
**TUGAS AKHIR –KI141502**

# OPTIMASI *MIXED INTEGER LINEAR PROGRAMMING* PADA PENGHINDARAN KONFLIK MENGGUNAKAN MODEL *VELOCITY AND ALTITUDE CHANGE*

**IZDIHAR FARAHDINA**
**NRP 5112 100 191**

**Dosen Pembimbing I**

**Victor Hariadi, S.Si., M.Kom.**

**Dosen Pembimbing II**

**Rully Soelaiman, S.Kom., M.Kom.**

**JURUSAN TEKNIK INFORMATIKA**
**FAKULTAS TEKNOLOGI INFORMASI**
**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**
**SURABAYA**
**2016**

*[Halaman ini sengaja dikosongkan]*

UNDERGRADUATE THESES – KI141502

# MIXED INTEGER LINEAR PROGRAMMING OPTIMIZATION IN CONFLICT AVOIDANCE USING VELOCITY AND ALTITUDE CHANGE MODEL

**IZDIHAR FARAHDINA**
**NRP 5112 100 191**

**First Supervisor**

**Victor Hariadi, S.Si., M.Kom.**

**Second Supervisor**

**Rully Soelaiman, S.Kom., M.Kom.**

**DEPARTMENT OF INFORMATICS**
**FACULTY OF INFORMATION TECHNOLOGY**
**SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY**
**SURABAYA**
**2016**

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

## OPTIMASI *MIXED INTEGER LINEAR PROGRAMMING* PADA PENGHINDARAN KONFLIK MENGGUNAKAN MODEL *VELOCITY AND ALTITUDE CHANGE*

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Dasar dan Terapan Komputasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
**IZDIHAR FARAHDINA**
NRP. 5112 100 191

**Disetujui oleh Pembimbing Tugas Akhir:**

1. Victor Hariadi, S.Si., M.Kom. .....
   NIP 19691228 199412 1 001 (Pembimbing 1)

2. Rully Soelaiman, S.Kom, M.Kom. .....
   NIP 19700213 199402 1 001 (Pembimbing 2)

**SURABAYA**
**JUNI, 2016**

*[Halaman ini sengaja dikosongkan]*

**OPTIMASI *MIXED INTEGER LINEAR PROGRAMMING* PADA PENGHINDARAN KONFLIK MENGGUNAKAN MODEL *VELOCITY AND ALTITUDE CHANGE***


**Nama Mahasiswa : Izdihar Farahdina**
**NRP : 5112100191**
**Jurusan : Teknik Informatika – FTIf ITS**
**Dosen Pembimbing I : Victor Hariadi, S.Si, M.Kom.**
**Dosen Pembimbing II : Rully Soelaiman, S.Kom., M.Kom.**



**Abstrak**

*Penghindaran konflik atau conflict avoindance (CA) adalah hal yang sangat penting dalam Air Traffic Management (ATM). Salah satu kriteria konflik adalah pesawat berada dalam protected zone (PZ). PZ adalah zona aman pesawat, yaitu 5 nm jarak horizontal minimum antara pesawat atau 1.000 ft sebagai jarak vertical minimum. Secara dasar conflict detection dan resolution (CDR) diselesaikan dengan menggunakan tiga manuver yaitu kecepatan, ketinggian, dan sudut arah terbang. Permasalahan utama CDR adalah memberikan solusi yang optimal tanpa mengganggu time flight yang telah diberikan.*

*Dalam tugas akhir ini akan diimplementasikan model velocity change and altitude (VAC) sebagai media CDR dan solusi untuk menghindari konflik antar pesawat. Model VAC adalah model yang menggunakan mixed integer linear programming (MILP). Model VAC memberikan dua solusi untuk menghindari pesawat dari konflik yaitu dengan perubahan level ketinggian dan kecepatan. Fungsi tujuan model VAC adalah meminimalkan perubahan ketinggian dan kecepatan agar pesawat tetap terbang sesuai time flight. Keluaran dari model VAC adalah ketinggian dan kecepatan pesawat yang tidak konflik dengan pesawat lainnya.*

*Berdasarkan hasil uji coba, untuk uji kebenaran dengan 10 pesawat yang dilakukan oleh penulis memberikan hasil sama benarnya dengan hasil uji coba yang dilakukan oleh Alonso et al [1]*

*dikarenakan semua konflik yang terjadi pada sebelum optimasi telah dapat dihindari. Sementar, hasil uji kinerja 37 pesawat menunjukkan nilai obyektif 6.5562 dengan jumlah constraint sebanyak 29130, 27602 variabel binary, dan 37 variabel continue. Dengan waktu cpu time 21:19,15 menit dan real time 21: 46,08 menit..*

**Kata kunci:** ***Conflict Avoindance , Velocity and Altitude Change, Air Traffic Management, Conflict Detection and Resolution, Mixed Integer Linear Programming.***

# MIXED INTEGER LINEAR PROGRAMMING OPTIMIZATION IN AIR CONFLICT AVOIDANCE USING VELOCITY AND ALTITUDE CHANGE MODEL


| | |
|---|---|
| **Name** | **: Izdihar Farahdina** |
| **NRP** | **: 5112100191** |
| **Department** | **: Teknik Informatika – FTIf ITS** |
| **Supervisor I** | **: Victor Hariadi, S.Si, M.Kom.** |
| **Supervisor II** | **: Rully Soelaiman, S.Kom., M.Kom.** |



## Abstract

*Conflict avoidance (CA) is an important thing in the Air Traffic Management (ATM). One of the conflict criteria is an aircraft inside the protected zone (PZ). PZ is safety zone for aircraft, which is 5 nm minimum horizontal distance between the aircraft or 1,000 ft as the minimum vertical distance. On the basis of conflict detection and resolution (CDR) were completed using three maneuver of speed, altitude and angle of flight direction. The main problem CDR is to provide an optimal solution without disturbing the time flight has been given.*

*In this Final Project will implement a velocity change and altitude (VAC) model as CDR media and solutions to avoid conflicts between aircraft. Model VAC is a model that uses mixed integer linear programming (MILP). VAC models provide two solutions to avoid aircrft from conflict is to change the level of altitude and speed. Objective function of VAC models is to minimize changes in altitude and speed to keep the aircraft flew appropriate time flight. Result of the model is the altitude and air speed that does not conflict with other aircraft .*

*Based on testing results, for corectness test of the 10 aircraft conducted by the authors give the results as true as the results of experiments performed by Alonso et al [1] because of all the conflicts that occurred in prior optimizations have been avoided. Meanwhile, the results of the performance test of the 37 aircraft is 6.5562*

*for objective value, 29 130 number of constraints, 27 602 binary variables and 37 variables continue with cpu time and real time was 21:19,15 minutes and 21: 46,08 minutes.*

***Keywords: Conflict Avoindance , Velocity and Altitude Change, Air Traffic Management, Conflict Detection and Resolution, Mixed Integer Linear Programming.***

# KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul

**"OPTIMASI *MIXED INTEGER LINEAR PROGRAMMING* PADA PENGHINDARAN KONFLIK DENGAN MODEL *VELOCITY AND ALTITUDE CHANGE*"**

Pengerjaan tugas akhir ini merupakan suatu kesempatan yang sangat berharga bagi penulis, karena dengan pengerjaan tugas akhir ini, penulis bisa memperdalam, meningkatkan, serta mengimplementasikan apa yang telah dipelajari penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya buku tugas akhir ini tidak terlepas dari bantuan dan dukungan semua pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Ayah, Ibu, Kakak dan keluarga yang selalu mendukung tiap pilihan dan mendoakan yang terbaik.
3. Bapak Rully Soelaiman S.Kom. M.Kom., selaku dosen pembimbing kedua yang telah memberikan inspirasi, kepercayaan, motivasi, bimbingan, dukungan, nasehat, perhatian, bantuan, serta nilai-nilai keteladan kepada penulis.
4. Bapak Victor Hariadi S.Si. M.Kom., selaku dosen pembimbing pertama yang telah memberikan kepercayaan, motivasi, bimbingan, dukungan, nasehat, perhatian, serta bantuan kepada penulis.
5. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang banyak memberikan ilmu dan bimbingan bagi penulis.
6. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.

7. Teman-teman ‘Rantau’ dan TC Angkatan 2012 yang selalu mendukung selama proses pengerjaan tugas akhir.
8. Serta semua pihak yang turut membantu penulis dalam menyelesaikan tugas akhir ini.

Sebagai manusia biasa, penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan dan memiliki banyak kekurangan. Sehingga dengan segala kerendahan hati, penulis memohon maaf atas tidak kesempurnaan itu.

Surabaya, 8 Juni 2016

Izdihar Farahdina

# DAFTAR ISI

LEMBAR PENGESAHAN ..... v
Abstrak ..... vii
Abstract ..... ix
KATA PENGANTAR ..... xi
DAFTAR ISI ..... xiii
DAFTAR GAMBAR ..... xv
DAFTAR TABEL ..... xvii
DAFTAR NOTASI ..... xxi
BAB I PENDAHULUAN ..... 1
1.1 Latar Belakang ..... 1
1.2 Rumusan Permasalahan ..... 3
1.3 Batasan Permasalahan ..... 3
1.4 Tujuan ..... 4
1.5 Manfaat ..... 4
1.6 Metodologi ..... 4
1.7 Sistematika Penulisan ..... 6
BAB II PEMBAHASAN MODEL OPTIMASI ..... 9
2.1 Tinjauan Pustaka ..... 9
2.1.1 Collision Avoidance ..... 9
2.1.2 Model Velocity and Altitude Change ..... 11
2.1.3 Struktur Pengambilan Keputusan pada Model VAC 12
2.2 Pembahasan Model ..... 14
2.2.1 Notasi ..... 14
2.2.2 Model VC ..... 14
2.2.3 Model AC ..... 25
2.2.4 Penggabungan Model VC dan AC ..... 29
BAB III PERANCANGAN PERANGKAT LUNAK ..... 31

3.1 Model VAC Secara Umum ..... 31
3.2 Perancangan Data ..... 31
3.2.1 Data Masukan ..... 32
3.2.2 Data Keluaran ..... 34
3.3 Perancangan dan *Pseudocode* ..... 36
3.3.1 Variabel yang Digunakan Pada Perancangan Penyelesaian Penghindaran Konflik Menggunakan Model VAC ..... 36
3.3.2 Perancangan Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC ..... 45
3.3.3 Perancangan Model VAC untuk Penyelesaian Masalah Penghindaran Konflik ..... 55
BAB IV IMPLEMENTASI ..... 61
4.1 Lingkungan Implementasi ..... 61
4.2 Implementasi ..... 61
4.2.1 Implementasi Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC ..... 62
4.2.2 Implementasi Model VAC untuk Penyelesaian Masalah Penghindaran Konflik ..... 69
BAB V HASIL UJI COBA DAN EVALUASI ..... 75
5.1 Lingkungan Pelaksanaan Uji Coba ..... 75
5.2 Data Uji Coba ..... 75
5.3 Skenario Uji Coba ..... 77
5.3.1 Uji Kinerja ..... 77
5.3.2 Uji Kebenaran ..... 80
BAB VI KESIMPULAN DAN SARAN ..... 87
6.1 Kesimpulan ..... 87
6.2 Saran ..... 88
DAFTAR PUSTAKA ..... 89
A LAMPIRAN A ..... 91
BIODATA PENULIS ..... 103

# DAFTAR GAMBAR

Gambar 2.1 Struktur Pengambilan Keputusan Operasi Conflict Avoidance dengan Model VAC ..... 13
Gambar 2.2 Konstruksi Geometri dari *Difference Vector* ..... 16
Gambar 2.3 Konstruksi Geometri *Conflict Avoidance* ..... 17
Gambar 2.4 Konstruksi Geometri Batas *Safety Distance* ..... 18
Gambar 2.5 Situasi *False Conflict* ..... 24
Gambar 2.6 Titik Potong Vector $v_f^1 \ dan \ v_f^2$ ..... 25
Gambar 2.7 Konstruksi Geometri *Head to Head* ..... 27
Gambar 3.1 Hasil Keluaran Variasi Kecepatan, Perubahan Level dan Level Pesawat setelah Optimasi ..... 34
Gambar 3.2 Level Pesawat setelah Optimasi ..... 35
Gambar 3.3 *Pseudocode* Sudut α dan *Similar Coordinates* ..... 46
Gambar 3.4 *Pseudocode* Sudut ω ..... 47
Gambar 3.5 *Pseudocode* Sudut $\hat{\omega}$ (Bagian 1) ..... 48
Gambar 3.6 *Pseudocode* Sudut $\hat{\omega}$ (Bagian 2) ..... 49
Gambar 3.7 *Pseudocode* Sudut $l, g, \hat{l}, dan \ \hat{g}$ ..... 49
Gambar 3.8 *Pseudocode* Deteksi *Head to Head* ..... 50
Gambar 3.9 *Pseudocode* Titik Potong ..... 51
Gambar 3.10 *Pseudocode* Deteksi Situasi *False Conflict* ..... 52
Gambar 3.11 *Pseudocode* Deteksi Kasus Anomali ..... 53
Gambar 3.12 *Pseudocode* Parameter Konstruksi Geometri ..... 54
Gambar 3.13 *Pseudocode* Model VAC (Bagian 1) ..... 55
Gambar 3.14 *Pseudocode* Model VAC (Bagian 2) ..... 56
Gambar 3.15 *Pseudocode* Model VAC (Bagian 3) ..... 57
Gambar 3.16 *Pseudocode* Model VAC (Bagian 4) ..... 58
Gambar 3.17 *Pseudocode* Model VAC (Bagian 5) ..... 59
Gambar 5.1 Representasi Data Masukan Dalam Microsoft Excel 76
Gambar 5.2 Visualisasi Data Uji Coba 37 Pesawat ..... 78
Gambar 5.3 Visualisasi Hasil Optimasi pada Uji Coba 37 Pesawat ..... 79
Gambar 5.4 Visualisasi Data Uji Coba 10 Pesawat ..... 81

Gambar 5.5 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 1) ..... 82
Gambar 5.6 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 2) ..... 83
Gambar 5.7 Visualisasi Hasil Uji Coba 10 Pesawat oleh Alonso 84
Gambar 5.8 Visualisasi Hasil Uji Coba 10 Pesawat oleh Penulis dengan Threshold 0.35 ..... 85
Gambar A.1 Hasil Uji Kinerja ..... 91
Gambar A.11 Visualisasi Hasil Uji Coba oleh Alonso ..... 94
Gambar A.12 *Problem Summary*, *Performance Information*, dan *Solution Summary* Uji Coba 37 Pesawat ..... 98
Gambar A.13 *Optimization Result* Variabel Variasi Kecepatan Uji Coba 10 Pesawat ..... 98
Gambar A.14 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 1) ..... 99
Gambar A.15 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 2) ..... 100
Gambar A.16 *Problem Summary*, *Performance Information*, dan *Solution Summary* Uji Coba 10 Pesawat ..... 101

# DAFTAR TABEL

Tabel 3.1 Contoh Data Masukan(Bagian 1)..... 32
Tabel 3.2 Contoh Data Masukan (Bagian 2)..... 33
Tabel 3.3 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 1)..... 37
Tabel 3.4 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 2)..... 38
Tabel 3.5 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 3)..... 39
Tabel 3.6 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 4)..... 40
Tabel 3.7 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 1)..... 41
Tabel 3.8 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 2)..... 42
Tabel 3.9 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 3)..... 43
Tabel 3.10 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 4)..... 44
Tabel 3.11 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 5)..... 45
Tabel 4.1 Lingkungan Implementasi Perangkat Lunak ..... 61
Tabel 5.1 Lingkungan Uji Coba..... 75
Tabel 5.2 Data 10 Pesawat..... 77
Tabel 5.4 *Optimization Result* Hasil Variasi Kecepatan Uji Coba 10 Pesawat ..... 81
Tabel A.1 Daftar Penjelasan Isi *Problem Summary* (Bagian 1)..... 95
Tabel A.2 Daftar Penjelasan Isi *Problem Summary* (Bagian 2)..... 96
Tabel A.3 Daftar Penjelasan Isi *Performance Information* ..... 96
Tabel A.4 Daftar Penjelasan Isi *Solution Summary* (Bagian 1)..... 96

Tabel A.5 Daftar Penjelasan Isi *Solution Summary* (Bagian 2).... 97

# DAFTAR KODE SUMBER

Kode Sumber 4.1 Implementasi Sudut α dan Similar Coordinates (Bagian 1)..... 62
Kode Sumber 4.2 Implementasi Sudut α dan *Similar Coordinates* (Bagian 2)..... 63
Kode Sumber 4.3 *Implementasi* Sudut ω..... 63
Kode Sumber 4.4 Implementasi Sudut $\hat{\omega}$ ..... 64
Kode Sumber 4.5 Implementasi Sudut $l, g, \hat{l}$, dan $\hat{g}$ ..... 65
Kode Sumber 4.6 Implementasi Deteksi *Head to Head* ..... 65
Kode Sumber 4.7 Implementasi Titik Potong..... 66
Kode Sumber 4.8 Implementasi Deteksi Situasi *False Conflict* (Bagian 1)..... 67
Kode Sumber 4.9 Implementasi Deteksi Situasi *False Conflict* (Bagian 2)..... 68
Kode Sumber 4.10 Implementasi Deteksi Kasus Anomali..... 68
Kode Sumber 4.11 Implementasi Parameter Konstruksi Geometri ..... 69
Kode Sumber 4.12 Implementasi Model VAC (Bagian 1)..... 70
Kode Sumber 4.13 Implementasi Model VAC (Bagian 2)..... 71
Kode Sumber 4.14 Implementasi Model VAC (Bagian 3)..... 72
Kode Sumber 4.15 Implementasi Model VAC (Bagian 4)..... 73
Kode Sumber 4.16 Implementasi Model VAC (Bagian 5)..... 74

*[Halaman ini sengaja dikosongkan]*

# DAFTAR NOTASI

| | |
|---|---|
| $\mathcal{F}$ | Himpunan pesawat-pesawat dalam sektor (1,...,*F*). |
| $\mathcal{Z}^f$ | Himpunan level ketinggian dalam sektor (1,...,Z). |
| $x_f, y_f$ | Koordinat dalam sektor untuk setiap pesawat (1,...,*f*). |
| $v_f$ | Konfigurasi kecepatan awal (sebelum optimasi) yang terdapat untuk setiap pesawat (1,...,*f*). |
| $\underline{v}_f, \overline{v}_f$ | Konfigurasi kecepatan minimum dan maksimum yang terdapat pada setiap pesawat (1,...,*f*). |
| $z_f$ | Ketinggian awal (sebelum optimasi) dalam sektor untuk setiap pesawat (1,...,*f*). |
| $r_f$ | *Safety radius*, biasanya bernilai 2,5 nm. *Safety radius* terpadat pada setiap pesawat (1,...,*f*). |
| $m_f^*$ | Sudut arah, terdapat pada setiap pesawat (1,...,*f*) dan bernilai (-π,π] |
| $c_f^{q^+}, c_f^{q^-}$ | *Cost* untuk positif dan negatif variasi perubahan kecepatan untuk setiap pesawat (1,...,*f*). |
| $c_f^j$ | *Cost* untuk jumlah perubahan level ketinggian pada setiap pesawat (1,...,*f*). |
| $w_n$ | Nilai berat untuk setiap *objective function* dengan nilai antara 0 dan 1. |
| $g_{ij}, l_{ij}$ | Sudut *safety distance* yang digunakan sebagai deteksi *collision* untuk setiap kombinasi pesawat $(i, j \in F)$ |
| $\alpha_{ij}$ | Sudut antara jarak dan *safety radius* yang digunakan untuk perhitungan sudut *safety distance* untuk setiap kombinasi pesawat $(i, j \in F)$. |
| $\omega_{ij}$ | Sudut antara garis yang terhubung antar dua pesawat dan sumbu horizontal yang digunakan untuk perhitungan sudut *safety distance* untuk setiap kombinasi pesawat $(i, j \in F)$. |
| $hth_{ij}$ | BINARY PARAMETER, bernilai 1 jika pesawat $i$ dan pesawat $j$ terbang menuju satu sama lain (*head to head*), 0 untuk sebaliknya. $(i, j \in F)$. |

| | |
|---|---|
| $sc_{ij}$ | BINARY PARAMETER, bernilai 1 jika pesawat $i$ dan pesawat $j$ memiliki jarak kurang dari *safety distance*, 0 untuk sebaliknya. $(i, j \in F)$. |
| $pc_{ij}$ | BINARY PARAMETER, bernilai 1 jika pesawat $i$ dan pesawat $j$ memiliki jarak pada sumbu horizontal kurang dari *safety distance*, 0 untuk sebaliknya. $(i, j \in F)$. |
| $fc_{ij}$ | BINARY PARAMETER, bernilai 1 jika pesawat $i$ dan pesawat $j$ terbang menjauhi satu sama lain, 0 untuk sebaliknya. $(i, j \in F)$. |
| $h_i, h_j, k_i, k_j$ | Parameter yang digunakan untuk meringkas batasan konstruksi geometri antara pesawat $i$ dan pesawat $j$ $(i, j \in F)$. |
| $h'_i, h'_j, k'_i, k'_j$ | Parameter yang digunakan untuk meringkas batasan konstruksi geometri jika pesawat $i$ dan pesawat $j$ termasuk dalam kasus anomali $(i, j \in F)$. |
| $\widehat{\omega}_{ij}$ | Sudut antara pesawat *i* dan pesawat *j* yang berdasarkan letak quadrant pesawat $j$ dengan pesawat $i$ sebagai pantauannya $(i, j \in F)$. |
| $\hat{g}_{ij}, \hat{l}_{ij}$ | Sudut yang berdasarkan $\widehat{\omega}_{ij} \pm \alpha_{ij}$, dilakukan untuk setiap pasangan pesawat, yaitu pesawat i dan pesawat $j$ $(i, j \in F)$. |
| $v_f^1 . v_f^2$ | Vector kecepatan untuk setiap pesawat (1,...,$f$). |
| $ip_{ij}$ | Titik potong antara pesawat *i* dan pesawat *j* $(i, j \in F)$. |
| $d_{ij}$ | Jarak eucludian antara pesawat *i* dan pesawat *j* $(i, j \in F)$. |
| $d_{ij}^1$ | Jarak antara titik potong $ip_{ij}$ dan posisi pesawat *i,* hal ini berlaku untuk pesawat *j* $(i, j \in F)$. |
| $d_{ij}^2$ | Jarak antara titik potong $ip_{ij}$ dan posisi pesawat *i* ditambah sudut $\left(x_i + \cos(m_i^*), y_i + sin(m_i^*)\right)$, hal ini berlaku untuk pesawat *j* $(i, j \in F)$. |

Variabel

$q_f, q_f^+, q_f^-$ REAL, variabel keputusan variasi kecepatan untuk setiap pesawat yang dibagi oleh dua *variable nonnegative* yaitu $q_f^+$ dan $q_f^-$. Dimana $q_f = q_f^+ - q_f^-$ dengan $q_f^+$ mengambil nilai positif variasi kecepatan dan $q_f^-$ mengambil nilai negative variasi kecepatan untuk setiap pesawat (1,...,*f*).

$\rho_f$ INTEGER, variabel keputusan yang mengambil nilai jumlah perubahan level ketinggian untuk setiap pesawat (1,...,*f*).

$z'_f$ INTEGER, variabel keputusan yang mengambil nilai level ketinggian untuk setiap pesawat (1,...,*f*).

$V_f^z$ BINARY, variabel keputusan yang bernilai 1 jika pesawat $f$ terbang pada level ketinggian $z$ dan 0 untuk sebaliknya. Dilakukan untuk setiap pesawat (1,...,*f*) dan pada setiap level ketinggian (1,...,*z*).

$\delta_{ijz}^n$ BINARY, variabel keputusan pada linearisasi *or-constraint* untuk model *velocity change* ($\delta_{ijz}^1, \delta_{ijz}^2, \delta_{ijz}^3, \delta_{ijz}^4$)dan pada model *altitude change* ($\delta_{ijz}^5$).

*[Halaman ini sengaja dikosongkan]*

# BAB I
# PENDAHULUAN

Pada bagian ini akan dijelaskan beberapa hal dasar mengenai tugas akhir ini meliputi latar belakang, rumusan permasalahan, batasan permasalahan, tujuan, manfaat dari tugas akhir, metodelogi serta sistematika penulisan tugas akhir.

## 1.1 Latar Belakang

Dengan ramainya pertumbuhan *traffic* di wilayah udara menambah kebutuhan yang harus diterapkan pada sistem *Air Traffic Control* (ATC). Penghindran konflik pesawat pun menjadi hal yang sangat serius untuk dibahas. Dalam bidang penerbangan atau lebih spesifik dibidang *Air Traffic Management* (ATM) lebih dikenal dengan *collision-conflict avoindance* (CA). Banyak inovasi tentang *conflict detection dan resolution* (CDR) yang telah diberikan dengan berbagai solusi dan pilihan manuver yang berbeda. Pada dasarnya solusi CA memakai tiga manuver yaitu kecepatan, ketinggian dan sudut arah terbang.

Permasalahan utamanya adalah bagaimana membuat sistem yang dapat menyediakan prosedur normal untuk membantu kontroler dan pilot ketika tindakan yang dilakukan gagal untuk menjaga pesawat dari konflik dan solusi tidak mengganggu *time flight* yang sudah disediakan. Konflik yang dimaksud adalah suatu peristiwa di mana dua atau lebih pesawat melanggar kriteria aman yang ditetapkan. Salah satu contoh kriteria adalah 5 nm jarak horizontal minimum antara pesawat atau 1.000 ft sebagai jarak vertical minimum. Kriteria ini disebut sebagai *protected zone* (PZ)

Tujuan dari permasalahan ini adalah menghindari pesawat dari konflik dan total perubahan yang tidak menggangu *time flight*. Dari tujuan tersebut, penulis mengangkat tema CA untuk tugas akhir ini. Penulis menggunakan referensi penyelesaian CA dari Alonso et al [1] , Pallotino et al [2], dan Frizzoli et al [3].

Solusi yang diberikan untuk permasalahan ini dengan menggunakan manuver kecepatan dan ketinggian. Model yang menggunakan solusi tersebut adalah model *velocity and altitude change* (VAC) yang diusulkan oleh Alonso et al [1] dengan menggunakan konstruksi geometri dan model *velocity change* (VC) yang diperkenalkan oleh Pallotino et al [2]. VAC bertujuan untuk mencari kecepatan dan ketinggian yang optimal agar pesawat terhindar dari kriteria konflik. Tujuan ini juga meminimalkan segala perubahan agar tidak mengubah *time flight*.

VC yang dijelaskan oleh Pallotino et al [2] adalah modifikasi dari ide perubahan kecepatan yang dijelaskan oleh Frizzoli et al [3]. Frizzoli et al [3] menjelaskan batasan variasi kecepatan yang tidak menggangu konfigurasi kecepatan pesawat. Variasi kecepatan ini yang akan ditambahkan dengan kecepatan awal sehingga membentuk kecepatan yang optimal.

Alonso et al [1] memodifikasi model VC yang dilakukan oleh Pallotino et al [2] dengan penambahan kriteria konflik yaitu kasus anomali dan *false conflict* serta memberikan alternatif solusi menggunakan manuver ketinggian. Resolusi ketinggian yang diusulkan oleh Alonso et al [1] dengan cara mengelompokan pesawat kedalam beberapa level ketinggian dimana jarak antar level adalah 1.000 ft. Penyelesaian model VAC menggunakan *Mixed Integer Linear Programming* (MILP).

*False conflict* adalah parameter yang mendeteksi jika dua pesawat terbang saling menjauhi satu sama lain dan kasus anomali adalah mendeteksi pembagi bernilai 0 pada konstruksi geometri. Pada model *altitude change* (AC) yang diusulkan oleh Alonso et al [1] dilakukan pendeteksian dua pesawat terbang menuju satu sama lain, hal ini disebut dengan deteksi situasi *head to head* dan pendeteksian pasangan pesawat memiliki jarak kurang dari *safety distance* atau disebut dengan deteksi *similar coordinates*. Pendeteksian *false conflict*, kasus anomali, *head to head*, dan *similar coordinates* bertujuan untuk menyederhanakan model VAC.

Dalam tugas akhir ini akan mengimplementasikan model yang diajukan oleh Antonio Alonso-Ayuso, Laureano F. Escudero,

dan F. Javier Martin-Campo dengan menggunakan data uji coba utama [1]. Hasil keluaran yang diperoleh dalam implementasi model VAC dapat dijadikan pertimbangan dalam melakukan keputusan yang tepat dalam penghindaran konflik yang terjadi.

## 1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Cara mendeteksi *false conflict*, kasus anomali, situasi *head to head*, dan *similar coordinates*.
2. Penerapan model VAC untuk mendapatkan kondisi pesawat yang tidak termasuk dalam kriteria konflik.
3. Melakukan implementasi model VAC untuk menyelesaikan permasalahan penghindaran konflik.
4. Melakukan uji coba model VAC untuk menyelesaikan permasalahan penghindaran konflik.

## 1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data uji coba yang digunakan adalah data uji coba yang diberikan oleh paper utama [2].
2. Penyelesaian masalah hanya memakai dua manuver yaitu kecepatan dan ketinggian.
3. Arah terbang dan kecepatan pesawat pada *time frame t* tetap untuk *time frame* selanjutnya, $t+1, t+2, t+3....t+n$ .
4. Wilayah pesawat dimasukan ke dalam sektor.
5. Model tidak mempertimbangkan hasil solusi dengan faktor luar, seperti cuaca, kecepatan angin, dan penggunaan bahan bakar.
6. *Safety radius* untuk setiap pesawat adalah sama, yaitu 2.5 nm. [4]
7. Koordinat, sudut arah terbang, kecepatan, kecepatan minimal, kecepatan maksimal, *safety radius*, dan penomoran

pesawat adalah variabel yang digunakan sebagai data awal untuk pengelolaan dan harus dimiliki untuk setiap pesawat.

## 1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Memahami desain model VAC untuk menyelesaikan permasalahan penghindaran konflik.
2. Mengimplementasikan model VAC untuk menyelesaikan permasalahan penghindaran konflik.
3. Mengevaluasi kinerja dari model VAC dengan melakukan uji coba.

## 1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu meminimalkan efek dari penghindaran konflik yaitu pesawat datang tidak sesuai jadwal (terlambat atau terlalu cepat). Sehingga semua pesawat dalam sektor memiliki kondisi pesawat yang optimal, yaitu optimal dalam kecepatan dan pemilihan level ketinggian.

## 1.6 Metodologi

Pembuatan tugas akhir ini terbagi menjadi beberapa tahap, tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Tahap Awal

   Tahap awal untuk memulai pengerjaan tugas akhir adalah pengidentifikasian masalah, pencarian studi literatur, perumussan masalah, dan penetapan tujuan penelitian. Studi literatur bertujuan untuk pencarian, pengumpulan, pembelajaran, dan pemahaman informasi dan literatur yang diperlukan untuk pembuatan aplikasi penyelesaian permasalahan penghindaran konflik. Dasar informasi yang diperlukan dalam pembuatan aplikasi diantaranya mengenai kriteria pesawat yang termasuk

dalam konflik, model VAC sebagai model untuk menyelesaikan permasalahan penghindaran konflik, dan struktur pengambilan keputusan penyelesaian penghindaran konflik. Informasi dan literatur didapatkan dari buku dan sumber-sumber informasi lain yang berhubungan.

2. Tahap Perancangan Aplikasi

Tahap ini meliputi perancangan aplikasi berdasarkan studi literatur dan pengumpulan data. Tahap ini mendefinisikan alur dari implementasi dan identifikasi terhadap data yang telah terkumpul. Pada tahapan ini dibuat rancangan dasar dari aplikasi yang akan dibuat. Serta dilakukan desain suatu aplikasi dan proses-proses yang ada.

3. Tahap Implementasi

Implementasi merupakan tahap membangun rancangan aplikasi yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan perancangan, sehingga menjadi sebuah aplikasi yang sesuai dengan apa yang direncanakan. Implementasi penyelesaian permasalahan penghindaran konflik dengan menggunakan model VAC. Implementasi penyelesaian permasalahan penghindaran konflik dengan menggunakan model VAC dilakukan dengan bantuan perangkat lunak SAS. Output yang diharapkan adalah kecepatan dan ketinggian yang optimal untuk setiap pesawat dan menghidari pesawat dari kriteria konflik.

4. Tahap Pengujian dan Evaluasi

Pada tahapan ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perancangan. Tahapan ini dimaksudkan juga untuk mengevaluasi jalannya aplikasi, mencari masalah yang mungkin timbul, dan mengadakan perbaikan jika terdapat kesalahan.

5. Tahap Penyusunan Buku tugas akhir

   Tahap ini merupakan penyusunan buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perancangan yang telah dibuat.

## 1.7 Sistematika Penulisan

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu rumusan permasalahan, batasan permasalahan, dan sistematika penulisan juga merupakan bagian dari bab ini.
2. Bab II. Tinjauan Pustaka & Pembahasan Model
   Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang untuk mendukung pembuatan tugas akhir ini dan memberi penjelasan mengenai permasalahan-permasalahan yang akan diselesaikan dengan menggunakan model-model dalam struktur pengambilan keputusan.
3. Bab III. Perancangan Perangkat Lunak
   Bab ini berisi penjelasan mengenai desain dan perancangan yang digunakan dalam tugas akhir, bahan dan peralatan yang digunakan untuk memenuhi tugas akhir, serta urutan pelaksanaan percobaan.
4. Bab IV. Implementasi
   Bab ini akan dilakukan pembuatan aplikasi yang dibangun dengan SAS 9.2 sesuai dengan permasalahan dan batasannya yang telah dijabarkan pada bab pertama.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisi penjelasan mengenai data hasil percobaan atau pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. Bab VI. Kesimpulan dan Saran
   Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan karya rancangan. Selain itu, pada bab ini diberikan saran-saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan tugas akhir.

*[Halaman ini sengaja dikosongkan]*

# BAB II
# PEMBAHASAN MODEL OPTIMASI

## 2.1 Tinjauan Pustaka

Bagian ini membahas tentang teori dasar yang menunjang penyusunan tugas akhir mengenai *Collision Avoidance* dan model *Velocity & Altitude Change*.

### 2.1.1 Collision Avoidance

*Collision-Conflict Avoidance* pada area udara adalah penghindaran konflik antara pesawat atau dengan faktor lainnya, CA terbagi oleh dua bagian yaitu *conflict-collision detection (CD)* dan *collision resolution (CR)*. Ada beberapa cara untuk memecahkan masalah *conflict avoidance* yaitu mengubah kecepatan, ketinggian dan perubahan sudut terbang.

*Conflict-Collision Detection (CD)* adalah proses pendeteksian konflik antara dua atau lebih pesawat atau antara pesawat dan beberapa kendala pada wilayah udara seperti wilayah udara terlarang, dan kondisi cuaca yang buruk pada suatu wilayah. Contoh: terdapat dua pesawat terbang pada bidang horizontal yang sama (di ketinggian yang sama). Setiap pesawat memiliki *flight plan, flight plan* terdiri dari urutan titik untuk setiap waktu *t* pada kecepatan *v* yang dimiliki setiap pesawat. Salah satu pesawat dapat menentukan kemungkinan terjadinya konflik, dimana dua pesawat akan berada pada jarak tidak aman. Jarak tidak aman antar pesawat adalah 5 nm diluar TRACON - Terminal Radar Approach Control – dan 3 nm didalam TRACON. Keputusan atau solusi untuk pemecahan permasalahan CD disebut CR.

Dengan ramainya pertumbuhan *traffic* di wilayah udara menambah kebutuhan yang harus diterapkan pada sistem. Telah banyak metode CDR yang diusulkan, biasanya metode yang diusulkan melalui serangkaian batasan dan penyederhanaan lingkungan.

Konflik adalah suatu peristiwa di mana dua atau lebih pesawat melanggar kriteria aman yang ditetapkan. Salah satu contoh kriteria adalah 5 nm jarak horizontal minimum antara pesawat atau 1.000 ft sebagai jarak vertical minimum. Kriteria tersebut dikenal dengan nama PZ. PZ atau volume ruang udara pada setiap pesawat yang tidak boleh dilanggar oleh kendaraan lain. PZ juga dapat didefinisikan sebagai wilayah yang jauh lebih kecil (misalnya, sebuah bola dengan diameter 500 ft) atau bahkan dalam hal parameter selain jarak (misalnya, waktu).

Seperti model *conflict avoidance* yang diusulkan oleb Frazzoli et al. [3] dalam hal kriteria konflik dan juga kriteria perubahan manuver kecepatan. Frazzoli menyederhanakan lingkungan menjadi ruang dua dimensi dan memberikan batasan-batasan yang terkait dengan PZ. Batasan ini bertujuan agar tidak terjadinya konflik antar pesawat.

Pada tugas akhir ini, membahas CDR dengan metode VAC dengan rujukan model yang diusulkan oleh Alonso et al. [1], yang merupakan pengembangan model VC oleh Pallotino et al. [2]. Pallottino et al. [2], mengusulkan pendekatan berdasarkan MILP untuk mendeteksi dan memecahkan masalah konflik. Pallotino et al. [2] menggunakan manuver kecepatan dan sudut terbang sebagai solusi dari permasalah penghindaran konflik. Dengan model yang diusulkan yaitu VC problem dan *heading angle change* (HAC). VC terdiri dari pesawat terbang di dengan arah sudut *m* dan hanya bisa mengubah kecepatan sekali dengan variasi kecepatan disebut $q_i$. Nilai $q_i$ dapat bernilai positif (akselerasi), negatif (deselerasi) atau nol (tidak ada variasi kecepatan). Variasi kecepatan dibatasi dengan batas atas kecepatan ($v_i$, *max*) dan batas bawah kecepatan ($v_i$, *min*). Kriteria konflik dan batas perubahan kecepatan yang diusulkan oleh Pallotino et al. [2] sama seperti yang model yang diusulkan oleh Frazzoli et al. [3]. Pallotino menambahkan pendekatan geometris untuk mempelajari kondisi non konflik dan mendefinisikan beberapa kendala untuk memperbaiki model Frazzoli et al. [3]. Model ini tidak selalu mendapatkan solusi untuk masalah konflik karena dalam kasus *head to head* perubahan kecepatan saja tidak cukup

untuk menghindari konflik. Model lainnya adalah HAC, model ini mengasumsikan bahwa kecepatan semua pesawat adalah sama dan masing-masing dapat mengubah arah terbang hanya sekali dengan perubahan sudut deviasi yang disebut *p*. Nilai *p* dapat bernilai positif (belok kiri), negatif (belok kanan) atau nol (tidak ada perubahan).

### 2.1.2 Model Velocity and Altitude Change

Model VAC adalah model VC dari Pallotino et al. [2] yang diperbaiki dan penambahan model AC. AC adalah model dengan menggunakan manuver ketinggian. Model yang diusulkan oleh Alonso et al. [1] ini, memungkinkan pesawat untuk terbang pada ketinggian yang berbeda dan mengubah kecepatan mereka untuk menghindari konflik. Beberapa perbaikan model VC yaitu penambahan situasi *false conflict* dan kasus anomali yang akan dibahas detail pada sub-bab 2.2.

Sejumlah pesawat dapat diperhitungkan dan masing-masing diberikan satu set parameter tetap (yaitu, koordinat, ketinggian, sudut dan jumlah perubahan ketinggian dan kecepatan). Tujuannya adalah menentukan level ketinggian penerbangan dan kecepatan optimal untuk setiap pesawat yang menjamin bahwa konflik telah dihindari.

Tujuan dari model ini terdiri dari menghindari semua konflik di sektor udara tertentu, meminimalkan jumlah kecepatan dan perubahan level. Model ini terbatas hanya pada satu sektor udara, karena akan diterapkan untuk waktu yang singkat dan akan membantu ATC pengambilan keputusan. Aspek ini akan juga berguna untuk mengurangi dimensi masalah, tetapi juga berlaku untuk seluruh wilayah udara. Menganalisis wilayah udara dibagi dalam sektor udara diusulkan untuk memecahkan masalah secara terpisah.

Model ini menghindari semua konflik dengan mengubah kecepatan dan menghindari beberapa kasus tidak layak di Pallottino et al. [2] dengan memperkenalkan manuver vertikal dengan perubahan ketinggian. Alonso et al. [1] memberikan kontribusi yang terdiri dari meningkatkan model dasar yang diusulkan oleh Pallotino

et al. [2] dan Frazzoli et al. [3] dengan memasukkan perubahan ketinggian untuk menghindari situasi tidak layak seperti situasi *head to head* yaitu dua peswat yang terbang menuju satu sama lain, dan pesawat yang terbang terlalu dekat satu sama lain.

Dalam langkah praproses dilakukan dengan mendeteksi pesawat yang akan berada dalam konflik dan memperbaiki beberapa variabel untuk meringkas perhitungan di model. Hal ini dilakukan untuk mengurangi waktu kerja pencarian solusi.

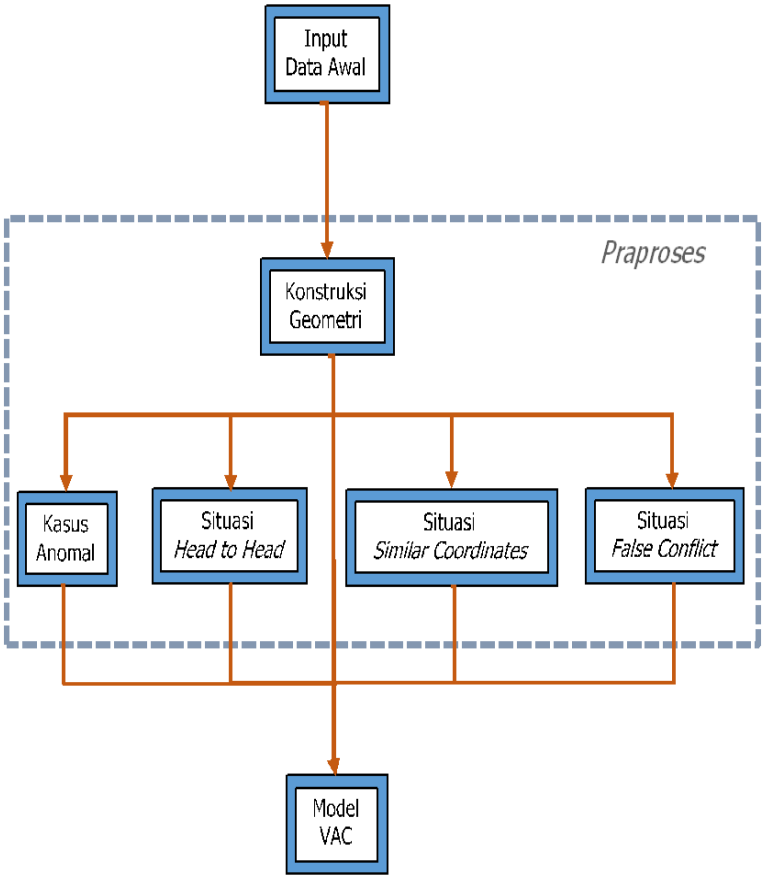
### 2.1.3 Struktur Pengambilan Keputusan pada Model VAC

Setiap pesawat *f* memiliki parameter koordinat (*x,y*), sudut (*m*) dengan batas (-π, π], kecepatan (*v*), *flight level* atau level ketinggian (*z*), dan *safety radius* (*r*), pada TA ini *safety radius* setiap pesawat adalah *safety distance*/2 atau 2,5 nm.

Dari data tersebut akan dihitung pengolahan data awal yang mendukung pengambilan keputusan pada model VAC seperti perhitungan konstruksi geometri, jarak antar pesawat *(d)*, *head to head (hth)*, *false conflict (fc)*, *similar coordinates (sc)*, *pathological case (pc)*, dan *intersection point (ip)*. Penjelasan parameter konstruksi geometri dapat dilihat pada sub-bab 2.2. Pendeteksian dua pesawat terbang menuju satu sama lain diinisialisasikan dengan parameter praproses *hth*. Untuk mendeteksi jika dua pesawat memiliki jarak kurang dari *safety distance* akan di hitung pada *sc*. *Intersection point (ip)* digunakan untuk menghitung mendeteksi *fc, fc* adalah parameter yang mendeteksi jika dua pesawat terbang saling menjauhi satu sama lain. Pada konstruksi geometri yang diusulkan memungkinkan adanya pembagi bernilai 0 oleh karena itu pendeteksian pembagi bernilai null akan disimpan pada parameter *pc*.

Setelah perhitungan praproses, akan dilakukan pencarian solusi menggunakan model VAC. Model VAC akan mencari variabel keputusan yaitu variasi kecepatan *q* untuk setiap pesawat $f$ yang terbagi menjadi dua variabel non-negatif $q^+$ dan $q^-$. $q = q^+ - q^-$ dimana $q^+$ dan $q^-$ adalah positif dan negatif variasi kecepatan dan variabel keputusan jumlah perubahan level $\rho$, nilai $\rho$ adalah non-

negatif integer. Batasan dan perumusan model VAC dapat dilihat pada sub-bab 2.2.




**Gambar 2.1 Struktur Pengambilan Keputusan Operasi Conflict Avoidance dengan Model VAC**

## 2.2 Pembahasan Model

Pada bagian ini akan membahas secara terperinci tentang model dari struktur pengambilan keputusan yang telah dijelaskan pada bab sebelumnya. Penjelasan pada model ini menyangkut beberapa proses yang harus dilalui dalam struktur pengambilan keputusan yang telah dijelaskan sebelumnya. perhitungan penyederhanaan konstruksi geometri, jarak antar pesawat *(d)*, *head to head (hth)*, *false conflict (fc)*, *similar coordinates (sc)*, *pathological case (pc)*, dan *intersection point (ip)*, pencarian solusi menggunakan model VAC. Konflik yang akan dideteksi adalah *false conflict* yaitu pasangan pesawat yang tidak termasuk konflik, *similar coordinates* yaitu pasangan pesawat yang memiliki jarak terlalu dekat atau kurang dari *safety distance*, dan *head to head* adalah pasangan pesawat yang terbang menuju satu sama lain.

### 2.2.1 Notasi

Untuk menjelaskan masing-masing model, berbagai notasi dan variabel keputusan yang digunakan dalam setiap model pada struktur pengambilan keputusan dapat dilihat pada daftar notasi.

### 2.2.2 Model VC

Model VC yang akan di jelaskan sesuai dengan VC model yang diusulkan oleh Pallotino et al. [2] dengan tambahan *anomalous case* dan *false conflict*. Model VC ini adalah bagian dari model VAC.

#### 2.2.2.1 Fungsi Obyektif VC

Pada bagian ini akan dijelaskan fungsi obyektif atau fungsi tujuan yang diinginkan pada model ini. Fungsi obyektif dari model VC ini adalah meminimalkan variasi kecepatan $q$ agar seminimal mungkin pesawat tidak terlambat ataupun terlebih dahulu sampai pada tujuan. Minimisasi $q$ menggunakan nilai absolut untuk meng-

hindari perubahan besar dalam rencana penerbangan awal dan di-normalisasi agar nilai diantara 0 dan 1, sehingga dapat dirumuskan sebagai berikut:

**Meminimalkan :**

$$\min \sum_{f \in F} |q_f| = \min \sum_{f \in F} \left( c_f^{q^+} q_f^+ + c_f^{q^-} q_f^- \right)$$

dinormalisasi menjadi

$$\text{VC min} \sum_{f \in F} \left( \frac{c_f^{q^+} q_f^+}{\bar{v}_f - \underline{v}_f} + \frac{c_f^{q^-} q_f^-}{\bar{v}_f - \underline{v}_f} \right) \tag{2.1}$$

$q_f^+$ dan $q_f^-$ akan memiliki nilai non-negatif. $q_f = q_f^+ - q_f^-$ dimana $q_f^+$ dan $q_f^-$ adalah nilai positif dan negatif dari $q$.

### 2.2.2.2 Batasan Variasi Kecepatan VC

Nilai dari variasi kecepatan $q$ harus berada diantara kecepatan minimum dan kecepatan maximum dari setiap pesawat oleh karena itu formula batasan variasi kecepatan adalah

$$\underline{v}_f \leq v_f + q_f \leq \bar{v}_f \tag{2.2 a}$$

seperti yang sudah dibahas dibagian sebelumnya, pada fungsi obyektif $q$ memiliki nilai non-negatif yaitu $q_f^+$ dan $q_f^-$. Formula untuk mendapatkan nilai non-negatif adalah

$$q = q^+ - q^- \tag{2.2 b}$$

### 2.2.2.3 Konstruksi Geometri VC

Penyederhaan konstruksi geometri ini digunakan pada batasan VC. Diberikat dua pesawat yang memiliki parameter masukan atau awal $(x_i, y_i, v_i, m_i^*)$ dan $(x_j, y_j, v_j, m_j^*)$, dengan parameter

koordinat (*x,y*), sudut (*m*) dengan batas (-π, π], kecepatan (*v*). dengan data tersebut dapat dirumuskan vector kecepatan untuk setiap pesawat, sebagai berikut:

$$\overrightarrow{v_f} = \begin{pmatrix} (v_f)\cos(m_f^*) \\ (v_f)\sin(m_f^*) \end{pmatrix} \tag{2.3}$$

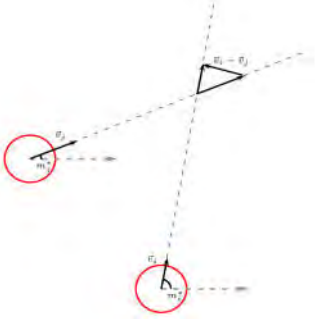
Untuk mendapatkan kecepatan optimal rumus diatas dapat diubah menjadi,

$$\overrightarrow{v_f} = \begin{pmatrix} (v_f + q_f)\cos(m_f^*) \\ (v_f + q_f)\sin(m_f^*) \end{pmatrix}$$

$q$ adalah variasi kecepatan yang digunakan untuk mendapatkan kecepatan optimal. Untuk setiap pasangan pesawat dapat dirumuskan *difference vector* yang merupakan *relative vector*, sebagai berikut:

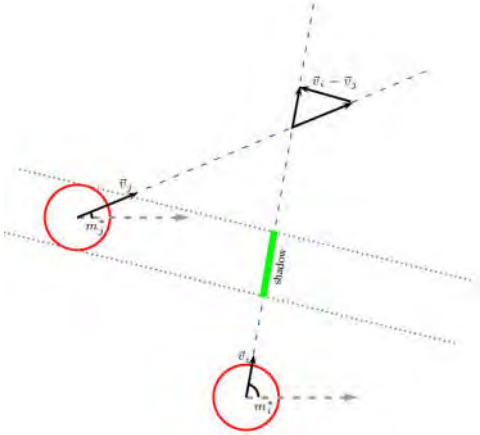
$$\overrightarrow{v_i} - \overrightarrow{v_j} = \begin{pmatrix} (v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \\ (v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*) \end{pmatrix} \tag{2.4}$$

dengan *difference vector* dapat diketahui apakah pasangan kedua pesawat terjadi konflik (lihat Gambar 2.2). Jika tan dari *difference vector* mengenai *safety distance* dari salah satu pesawat maka pasangan kedua pesawat tersebut mengalami konflik.

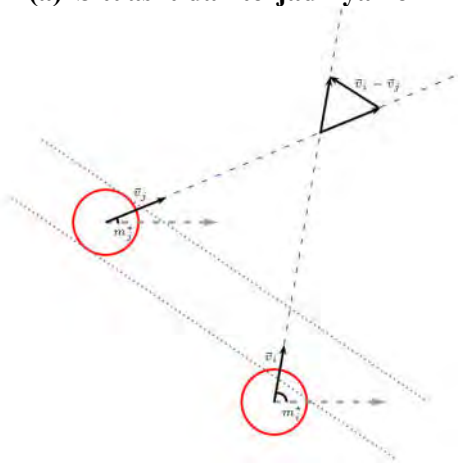


**Gambar 2.2 Konstruksi Geometri dari *Difference Vector***

Area tan dari *difference vector* dapat disebut sebagai area bayangan atau *shadow*. Dapat dilihat pada Gambar 2.3 (a) untuk *shadow* tidak mengenai *safety distance* pesawat *i* atau tidak terjadinya konflik dan Gambar 2.3 (b) menunjukkan kedua pasangan pesawat terjadi konflik dikarena *shadow* mengenai *safety distance* pesawat *i*.




**(a) Situasi tidak terjadinya konflik**



**(b) Situasi terjadinya konflik**

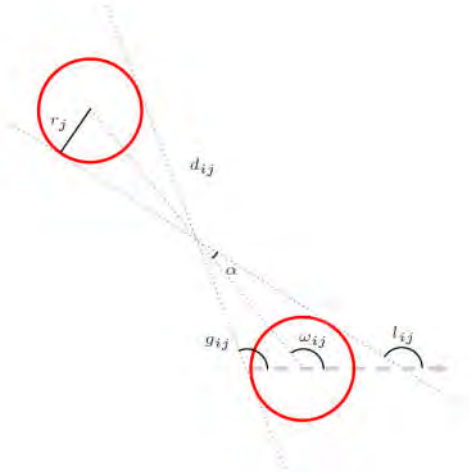
**Gambar 2.3 Konstruksi Geometri *Conflict Avoidance***

Untuk mengetahui konstruksi geometri dari *safety distance* atau jarak aman dapat dilihat pada Gambar 2.4. terdapat dua sudut utama sebagai pembetukan batas *safety distance* yaitu $\alpha$ dan $\omega$. Dimana $\alpha$ sebagai sudut antara *safety distance* dan jarak pesawat. *Safety distance* adalah *safety radius* dari pesawat *i* dan pesawat *j* dan jarak pesawat adalah $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, sehingga dapat dirumuskan sebagai berikut:

$$\alpha_{ij} = arcsin\left(\frac{r_i + r_j}{d_{ij}}\right) \qquad (2.5\ a)$$

Dan $\omega$ sebagai sudut yang menhubungkan kedua pesawat, sehingga dapat dirumuskan sebagai berikut:

$$\omega_{ij} = arcsin\left(\frac{y_i - y_j}{x_i - x_j}\right) \qquad (2.5\ b)$$




**Gambar 2.4 Konstruksi Geometri Batas *Safety Distance***

Jika $y_i = y_j$ maka $\omega_{ij} = 0$ dan jika $x_i = x_j$ maka $\omega_{ij} = \pi/2$

Sehingga untuk perumusan batas luar *safety distance* atau sudut $l$ sudut $g$ adalah

$$l_{ij} = \omega_{ij} + \alpha_{ij} \qquad (2.6\ a)$$
$$g_{ij} = \omega_{ij} - \alpha_{ij} \qquad (2.6\ b)$$

Perhitungan sudut $\alpha$, $\omega$, $l$, dan $g$ dilakukan pada saat praproses. Jika dilihat kembali pada Gambar 2.3 maka didapatkan kondisi untuk tidak terjadinya konflik adalah

$$\tan\left(\vec{v_i} - \vec{v_j}\right) \geq \tan(l_{ij})$$

atau

$$\tan\left(\vec{v_i} - \vec{v_j}\right) \leq \tan(g_{ij})$$

sehingga batasan penghidaran konflik adalah

$$\frac{(v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*)}{(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)} \geq \tan(l_{ij}) \qquad (2.7\ a)$$

atau

$$\frac{(v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*)}{(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)} \leq \tan(g_{ij}) \qquad (2.7\ b)$$

Dari batasan (2.7) terdapat 2 kondisi yaitu pembagi lebih besar dari 0 atau lebih kecil dari 0, sehingga batasan (2.7) dapat dirumuskan kembali menjadi

Jika $(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \geq 0$

$$(v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*) \geq \tan(l_{ij})\left((v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)\right)$$
$$(v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*) \geq (v_i + q_i)\tan(l_{ij})\cos(m_i^*) - (v_j + q_j)\tan(l_{ij})\cos(m_j^*)$$
$$(v_j + q_j)\tan(l_{ij})\cos(m_j^*) - (v_j + q_j)\sin(m_j^*) \geq (v_i + q_i)\tan(l_{ij})\cos(m_i^*) - (v_i + q_i)\sin(m_i^*)$$
$$(v_j + q_j)\underbrace{\left(\tan(l_{ij})\cos(m_j^*) - \sin(m_j^*)\right)}_{h_j} \geq (v_i + q_i)\underbrace{\left(\tan(l_{ij})\cos(m_i^*) - \sin(m_i^*)\right)}_{h_i}$$
$$(v_j + q_j)h_j \geq (v_i + q_i)h_i$$

$$(v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*) \leq \tan(g_{ij})\left((v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)\right)$$
$$(v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*) \leq (v_i + q_i)\tan(g_{ij})\cos(m_i^*) - (v_j + q_j)\tan(g_{ij})\cos(m_j^*)$$
$$(v_j + q_j)\tan(g_{ij})\cos(m_j^*) - (v_j + q_j)\sin(m_j^*) \leq (v_i + q_i)\tan(g_{ij})\cos(m_i^*) - (v_i + q_i)\sin(m_i^*)$$

$$(v_j + q_j)\underbrace{\left(\tan(g_{ij})\cos(m_j^*) - \sin(m_j^*)\right)}_{k_j} \leq (v_i + q_i)\underbrace{\left(\tan(g_{ij})\cos(m_i^*) - \sin(m_i^*)\right)}_{k_i}$$
$$(v_j + q_j)k_j \leq (v_i + q_i)k_i$$

Jika $(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) < 0$

$$\tan(l_{ij})\left((v_i + f_i)\cos(m_i^*) - (v_j + f_j)\cos(m_j^*)\right) \geq (v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*)$$
$$(v_i + q)\tan(l_{ij})\cos(m_i^*) - (v_j + q_j)\tan(l_{ij})\cos(m_j^*) \geq (v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*)$$
$$(v_i + q_i)\tan(l_{ij})\cos(m_i^*) - (v_i + q_i)\sin(m_i^*) \geq (v_j + q_j)\tan(l_{ij})\cos(m_j^*) - (v_j + q_j)\sin(m_j^*)$$
$$(v_i + q_i)\underbrace{\left(\tan(l_{ij})\cos(m_i^*) - \sin(m_i^*)\right)}_{h_i} \geq (v_j + q_j)\underbrace{\left(\tan(l_{ij})\cos(m_j^*) - \sin(m_j^*)\right)}_{h_j}$$
$$(v_i + q_i)h_i \geq (v_j + q_j)h_j$$

$$\tan(g_{ij})\left((v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)\right) \leq (v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*)$$
$$(v_i + q_i)\tan(g_{ij})\cos(m_i^*) - (v_j + q_j)\tan(g_{ij})\cos(m_j^*) \leq (v_i + q_i)\sin(m_i^*) - (v_j + q_j)\sin(m_j^*)$$
$$(v_i + q_i)\tan(g_{ij})\cos(m_i^*) - (v_i + q_i)\sin(m_i^*) \leq (v_j + q_j)\tan(g_{ij})\cos(m_j^*) - (v_j + q_j)\sin(m_j^*)$$
$$(v_i + q_i)\underbrace{\left(\tan(g_{ij})\cos(m_i^*) - \sin(m_i^*)\right)}_{k_i} \leq (v_j + q_j)\underbrace{\left(\tan(g_{ij})\cos(m_j^*) - \sin(m_j^*)\right)}_{k_j}$$
$$(v_i + q_i)k_i \leq (v_j + q_j)k_j$$

Dari penjelasan diatas menyatakan bahwa $h_i$, $h_j$, $k_i$, dan $k_j$ adalah sebuah parameter yang dapat dilakukan pada saat praproses dan ringkasan dari penjabaran formula diatas adalah

Kasus : $(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \leq 0$

$$\begin{cases} (v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \leq 0 \\ -(v_i + q_i)h_i + (v_j + q_j)h_j \leq 0 \end{cases} \quad (2.8\ a)$$

atau

$$\begin{cases} (v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \leq 0 \\ (v_i + q_i)k_i - (v_j + q_j)k_j \leq 0 \end{cases} \quad (2.8\ b)$$

Kasus : $(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \geq 0$

$$\begin{cases} -(v_i + q_i)\cos(m_i^*) + (v_j + q_j)\cos(m_j^*) \leq 0 \\ (v_i + q_i)h_i - (v_j + q_j)h_j \leq 0 \end{cases} \quad (2.8\ c)$$

atau

$$\begin{cases} -(v_i + q_i)\cos(m_i^*) + (v_j + q_j)\cos(m_j^*) \leq 0 \\ -(v_i + q_i)k_i + (v_j + q_j)k_j \leq 0 \end{cases} \quad (2.8\ d)$$

Formula (2.8) harus diubah menjadi batasan yang dapat digunakan pada model optimasi, sehingga formulasi batasan menjadi

$$(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \le M_1(1 - \delta_{ijz}^1) \quad (2.9\ a)$$
$$-(v_i + q_i)h_i + (v_j + q_j)h_j \le M_2(1 - \delta_{ijz}^1) \quad (2.9\ b)$$
$$(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*) \le M_3(1 - \delta_{ijz}^2) \quad (2.9\ c)$$
$$(v_i + q_i)k_i - (v_j + q_j)k_j \le M_4(1 - \delta_{ijz}^2) \quad (2.9\ d)$$
$$-(v_i + q_i)\cos(m_i^*) + (v_j + q_j)\cos(m_j^*) \le M_5(1 - \delta_{ijz}^3) \quad (2.9\ e)$$
$$(v_i + q_i)h_i - (v_j + q_j)h_j \le M_6(1 - \delta_{ijz}^3) \quad (2.9\ f)$$
$$-(v_i + q_i)\cos(m_i^*) + (v_j + q_j)\cos(m_j^*) \le M_7(1 - \delta_{ijz}^4) \quad (2.9\ g)$$
$$-(v_i + q_i)k_i + (v_j + q_j)k_j \le M_8(1 - \delta_{ijz}^4) \quad (2.9\ h)$$
$$\delta_{ijz}^1 + \delta_{ijz}^2 + \delta_{ijz}^3 + \delta_{ijz}^4 = 1 \quad (2.9\ i)$$

Dengan $M_1 = M_3 = M_5 = M_7 = (\overline{v_i} + \overline{v_j})$ adalah batas atas dari

$$(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)$$
$$(v_i + q_i)\cos(m_i^*) - (v_j + q_j)\cos(m_j^*)$$
$$-(v_i + q_i)\cos(m_i^*) + (v_j + q_j)\cos(m_j^*)$$
$$-(v_i + q_i)\cos(m_i^*) + (v_j + q_j)\cos(m_j^*)$$

$M_2 = M_6 = \overline{v_i}|h_i| + \overline{v_j}|h_j|$ adalah batas atas dari

$$-(v_i + q_i)h_i + (v_j + q_j)h_j$$
$$(v_i + q_i)h_i - (v_j + q_j)h_j$$

$M_4 = M_8 = \overline{v_i}|h_i| + \overline{v_j}|h_j|$ adalah batas atas dari

$$(v_i + q_i)k_i - (v_j + q_j)k_j$$
$$-(v_i + q_i)k_i + (v_j + q_j)k_j$$

$\delta_{ijz}^n$ adalah sebagai variabel keputusan, *i* dan *j* menginisialisasi pasangan pesawat dan *z* menginisialisasi ketinggian dikarena VAC adalah model yang memiliki dimensi ketinggian. Nilai sum dari $\delta_{ijz}^n$ bernilai 1 dikarenakan kasus pembagi lebih dari 0 dan kurang dari 0 adalah kasus yang terpisah dan hanya salah satu batasan yang

bernilai 1. $\delta_{ijz}^{1}$ adalah perubahan dari formula (2.8 a), $\delta_{ijz}^{2}$ adalah perubahan dari formula (2.8 b), $\delta_{ijz}^{3}$ adalah perubahan dari formula (2.8 c), dan $\delta_{ijz}^{4}$ adalah perubahan dari formula (2.8 d). Parameter $h_i$, $h_j$, $k_i$, dan $k_j$ adalah

$$h_i = \tan(l_{ij})\cos(m_i^*) - \sin(m_i^*) \quad (2.10\ a)$$
$$h_j = \tan(l_{ij})\cos(m_j^*) - \sin(m_j^*) \quad (2.10\ b)$$
$$k_i = \tan(g_{ij})\cos(m_i^*) - \sin(m_i^*) \quad (2.10\ c)$$
$$k_j = \tan(g_{ij})\cos(m_j^*) - \sin(m_j^*) \quad (2.10\ d)$$

#### 2.2.2.4 Kasus Anomali VC

Pada bagian konstruksi geometri dijelaskan jika pembagi bernilai lebih dari 0 atau kurang dari 0 . pada bagian kasus anomali akan membahas jika pembagi bernilai = 0. Solusi untuk hal ini adalah membahkan $\frac{\pi}{2}$ radian pada arah sudut *m*, suduh *l* dan sudut *g*. Mendeteksi pasangan pesawat yang memiliki kasus anomali dapat dilakukan pada praproses yaitu

$$pc_{ij} = \begin{cases} 1, \text{if } |x_i - x_j| \le r_i + r_j \\ 0 \end{cases} \quad (2.11)$$

Batasan (2.8) berubah dengan adanya penambahan parameter $pc_{ij}$ dan pengubahan sudut *m*, sudut *l*, dan sudut *g*. pengubahan pada batasan (2.9 a) akan menjadi

$$\begin{aligned}(v_i + q_i)\big(\cos(m_i^*)\,(1 - pc_{ij}) - sin(m_i^*)pc_{ij}\big) \\ - (v_j + q_j)\big(\cos(m_j^*)\,(1 - pc_{ij}) - sin(m_j^*)pc_{ij}\big) \\ \le M_1\big(1 - \delta_{ijz}^{1}\big)\end{aligned}$$

dengan $M_1$ tetap yaitu $(\overline{v_i} + \overline{v_j})$. Pada batasan (2.9 b) berubah mejadi

$$\begin{aligned}-(v_i + q_i)\big(h_i(1 - pc_{ij}) + h_i' pc_{ij}\big) + (v_j + q_j)(h_j(1 - pc_{ij}) + h_j' pc_{ij}) \\ \le M_2\big(1 - \delta_{ijz}^{1}\big)\end{aligned}$$

dengan $M_2$ menjadi

$$(\overline{v}_i|h_i| + \overline{v}_j|h_j|)(1 - pc_{ij}) + (\overline{v}_i|h_i'| + \overline{v}_j|h_j'|)pc_{ij}$$

Pola pengubahan pada batasan (2.9 a) dan (2.9 b) juga berlaku pada semua batasan (2.9). Pada parameter $h_i$, $h_j$, $k_i$, dan $k_j$ berubah menjadi

$$h_i' = -\tan\left(l_{ij} + \frac{\pi}{2}\right)\sin(m_i^*) - \cos(m_i^*) \quad (2.12\ a)$$

$$h_j' = -\tan\left(l_{ij} + \frac{\pi}{2}\right)\sin(m_j^*) - \cos(m_j^*) \quad (2.12\ b)$$

$$k_i' = -\tan\left(g_{ij} + \frac{\pi}{2}\right)\sin(m_i^*) - \cos(m_i^*) \quad (2.12\ c)$$

$$k_j' = -\tan\left(g_{ij} + \frac{\pi}{2}\right)\sin(m_j^*) - \cos(m_j^*) \quad (2.12\ d)$$

Parameter $h_i$, $h_j$, $k_i$, dan $k_j$ adalah parameter yang dilakukan pada saat praproses untuk mengurangi beban perhitungan pada saat model optimasi dijalankan.

### 2.2.2.5 False Conflict VC

False conflict adalah keadaan dimana dua pesawat terbang menjauhui titik potongnya (lihat Gambar 2.5). Keadaan ini seharusnya tidak terjadi konflik tetapi konstruksi geometri (2.7) menganggap kedua pesawat ini dalam keadaan konflik. Untuk mendeteksi apakah kedua pesawat termasuk dalam false conflict dapat dilakukan pada praproses dengan cara menghitung jarak pesawat ke titik potong dan jarak sudut ke titik potong. Formulanya adalah sebagai berikut

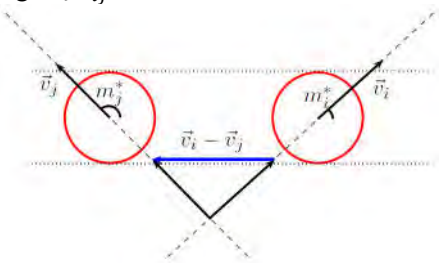
$$fc_{ij} = \begin{cases} 1, \text{if } d_{ij}^2 - d_{ij}^1 > 0 \ \wedge \ d_{ji}^2 - d_{ji}^1 > 0 \ \wedge sc_{ij}, hth_{ij} \neq 1 \\ 0 \end{cases} \quad (2.13)$$

dimana

$$d_{ij}^1 = jarak\{(x_i, y_i), ip_{ij}\} \quad (2.14\ a)$$

$$d_{ji}^{1} = jarak\{(x_j, y_j), ip_{ij}\} \quad (2.14\ b)$$
$$d_{ij}^{2} = jarak\{(x_i + \cos(m_i^*), y_i + sin(m_i^*)), ip_{ij}\} \quad (2.14\ c)$$
$$d_{ji}^{2} = jarak\{(x_j + \cos(m_j^*), y_j + sin(m_j^*)), ip_{ij}\} \quad (2.14\ d)$$

Parameter $d_{ij}^2, d_{ji}^2, d_{ij}^1, d_{ji}^1$ dilakukan pada saat praproses untuk mendukung perhitungan $fc_{ij}$.




**Gambar 2.5 Situasi *False Conflict***

$ip_{ij}$ adalah parameter titik potong yang terdiri dari $Xip_{ij}$ sebagai koordinat x dari titik potong dan $Yip_{ij}$ sebagai koordinat y dari titik potong. Perhitungan $ip_{ij}$ dilakukan pada saat praproses dan formula $ip_{ij}$ dapat dilihat pada sub-bab 2.2.2.5.1.

Penggunaan $sc_{ij}$ dan $hth_{ij}$ dikarenakan perhitungan jarak tersebut menghasilkan semua pasangan pesawat yang tidak konflik tetapi pada kasus false conflict data yang dibutukan hanya pasawat-pesawat yang terbang saling menjauhi titik potongnya. $sc_{ij}$ adalah parameter yang mendeteksi jika ada dua pesawat yang memiliki jarak kurang dari *safety distance* dan $hth_{ij}$ adalah parameter yang mendeteksi pasangan pesawat yang terbang menuju satu sama lain. $sc_{ij}$ dan $hth_{ij}$ akan dibahas pada model AC sub-bab 2.2.3.

Penggunaan $fc_{ij}$ sebagai batasan atau kriteria untuk model VC sehingga yang dihitung pada model VC adalah $fc_{ij} = 0$.

#### 2.2.2.5.1 Perhitungan Titik Potong

Untuk menghitung titik potong dapat menggunakan *velocity vector* (2.3) $\overrightarrow{v_f} = (v_f^1, v_f^2)$ dan koordinat $(x_f, y_f)$ dari setiap pesawat dengan menggunakan persamaan garis.

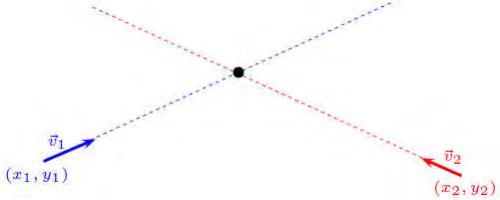
$$\frac{x - x_f}{v_f^1} = \frac{y - y_f}{v_f^2}$$

dimana

$$y = \frac{v_f^2}{v_f^1}x + (y_f - \frac{v_f^2}{v_f^1}x_f)$$

Sehingga jika $i$ dan $j$ adalah pasangan pesawat, titik potongnya $(Xip_{ij}, Yip_{ij})$ adalah

$$\left( \frac{y_j - \frac{v_j^2}{v_j^1}x_j - y_i - \frac{v_i^2}{v_i^1}x_i}{\frac{v_i^2}{v_i^1} - \frac{v_j^2}{v_j^1}}, \frac{\frac{v_i^2}{v_i^1}\left(y_j - \frac{v_j^2}{v_j^1}x_j\right) - \frac{v_j^2}{v_j^1}\left(y_i + \frac{v_i^2}{v_i^1}x_i\right)}{\frac{v_i^2}{v_i^1} - \frac{v_j^2}{v_j^1}} \right) \quad (2.15)$$




**Gambar 2.6 Titik Potong Vector $v_f^1$ dan $v_f^2$**

### 2.2.3 Model AC

Model AC adalah sub-bagian dari model VAC yang membahas tentang perubahan *altitude*. Pada bagian ini akan dijelaskan batasan-batasan AC, *similar coordinates* dan *head to head*.

### 2.2.3.1 Fungsi Obyektif AC

Pada bagian ini akan dijelaskan fungsi obyektif atau fungsi tujuan yang diinginkan pada model ini. Fungsi obyektif dari model AC ini adalah meminimalkan perubahan level atau ketinggian $\rho$ agar seminimal mungkin pesawat yang mengubah ketinggian atau level

**Meminimalkan :**

$$\text{AC min} \sum_{f \in F} c_f^j \rho_f \tag{2.16}$$

### 2.2.3.2 Batasan Ketinggian

Setiap pesawat harus berada di salah satu level ketinggian. Formula untuk batasan ini adalah pesawat harus berada di salah satu level ketinggian yang ada oleh karena itu formula batasan ketinggian adalah

$$\sum_{z \in Z^f} v_f^z = 1 \tag{2.17}$$

### 2.2.3.3 Head to Head

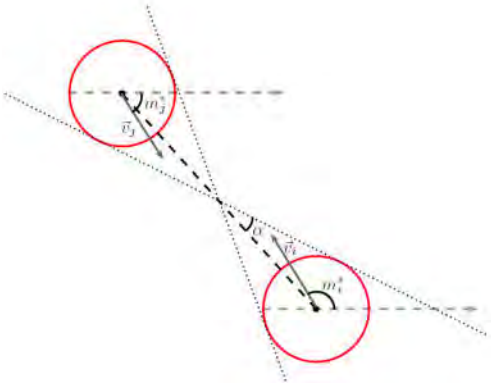
$hth_{ij}$ adalah parameter yang mendeteksi pasangan pesawat yang terbang menuju satu sama lain. $hth_{ij}$ dilakukan pada praproses. Pada pesawat yang terbang menuju satu sama lain tidak bisa di-selesaikan dengan solusi perubahan kecepatan dikarenakan kedua pesawat akan saling menabarak satu sama lain. Oleh karena solusi yang di berikan adalah dengan mengubah ketinggian salah satu pesawat. Untuk mendeteksi pesawat terbang menuju satu sama lain dengan menggunakan sudut $\omega$ (2.5 b). Formula $hth_{ij}$ adalah

$$hth_{ij} = \begin{cases} 1, \text{if } \hat{g}_{ij} \le m_i^* \le \hat{l}_{ij} \wedge \hat{g}_{ji} \le m_j^* \le \hat{l}_{ji} \\ 0 \end{cases} \tag{2.18}$$

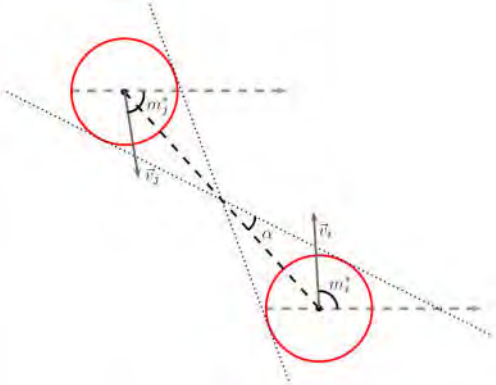
dimana

$$\hat{g}_{ij} = \hat{\omega}_{ij} - \alpha_{ij} \quad (2.19\ a)$$
$$\hat{l}_{ij} = \hat{\omega}_{ij} + \alpha_{ij} \quad (2.19\ b)$$

$\hat{\omega}_{ij}$ adalah $\omega_{ij}$ dengan tambahan kriteria dari letak quadran pesawat $j$ dilihat dari pesawat $i$. Jika pesawat $j$ berada di quadran I dan IV maka $\hat{\omega}_{ij} = \omega_{ij}$, jika berada di quadran II maka $\hat{\omega}_{ij} = \omega_{ij} + \pi$ dan untuk pesawat $j$ yang berada di quadran III $\hat{\omega}_{ij} = \omega_{ij} - \pi$.



**(a) Situasi *Head to Head***



**(b) Situasi tidak terjadi *Head to Head***

**Gambar 2.7 Konstruksi Geometri *Head to Head***

Perhitungan paramater $\widehat{\omega}_{ij}, \hat{g}_{ij}$, dan $\hat{l}_{ij}$ dilakukan pada saat praproses dikarenakan diperlukan untuk mendukung hasil parameter $hth_{ij}$.

### 2.2.3.4 Similar Coordinates

$sc_{ij}$ adalah paramaeter yang mendeteksi jika ada dua pesawat yang memiliki jarak kurang dari *safety distance*. Permasalahan ini diselesaikan dengan model AC. Perhitungan $sc_{ij}$ dilakukan pada saat praproses. Formula $sc_{ij}$ adalah

$$sc_{ij} = \begin{cases} 1, \text{if } d_{ij} \leq r_i + r_j \\ 0 \end{cases} \quad (2.20)$$

### 2.2.3.5 Batasan Perbedaan Ketinggian

Hasil praproses untuk parameter $hth_{ij}$ dan $sc_{ij}$ akan disimpan di variabel keputusan $\delta^5_{ijz}$. $\delta^5_{ijz}$ adalah variabel keputusan yang berperan sebagai penanda bahwa pasangan tersebut termasuk dalam model AC dan kedua pesawat harus memiliki level ketinggian yang berbeda, sehingga batasan tersebut di formulakan menjadi

$$\delta^5_{ijz} = 1 \qquad if\ hth_{ij} + sc_{ij} \geq 1 \quad (2.21)$$

Jika salah satu pasangan pesawat termasuk dalam kriteria *head to head* atau *similar coordinates* maka $\delta^5_{ijz}$ akan bernilai 1 (2.21). Untuk pasangan yang memiliki $\delta^5_{ijz}$ bernilai 1 akan harus terbang pada level yang berbeda, dinyatakan oleh formula

$$v_i^z + v_j^z \geq \delta^5_{ijz} - 1 \quad (2.22\ a)$$
$$v_i^z + v_j^z \leq 2 - \delta^5_{ijz} \quad (2.22\ b)$$

Jika kedua pesawat terbang pada level yang berbeda maka $\delta^5_{ijz} = 1$, $v_i^z + v_j^z \leq 1$ dan jika terbang pada level yang sama, $v_i^z + v_j^z \geq 2$ untuk $\delta^5_{ijz} = 0$. Untuk batasan (2.22) jika $\delta^5_{ijz} = 1$ maka pada (2.21

b) akan bernilai 1 dan jika $\delta_{ijz}^{5} = 0$ pada (2.21 a) lebih besar dari -1 dan (2.21 b) lebih kecil dari 2, sehingga batasan tidak terbatas karena jika $\delta_{ijz}^{5} = 0$ memiliki tiga kemungkinan, tidak berada di level yang sama, salah satu pesawat berbeda level dan kedua pesawat berada di level yang sama.

#### 2.2.3.6 Batasan Perubahan Level

Perubahan ketinggian level tidak dibatasi. Pesawat bisa untuk turun ataupun naik level melebihi dari 1 level. Untuk melihat berapa nilai perubahan level dapat dinyatakan seperti berikut

$$\sum_{z \,\epsilon\, Z^f} z v_f^z - z_f \leq \rho_f \qquad (2.23\ a)$$

$$z_f - \sum_{z \,\epsilon\, Z^f} z v_f^z \leq \rho_f \qquad (2.23\ b)$$

Nilai $\rho_f$ bersifat non-negatif integer.

$$\sum_{z \,\epsilon\, Z^f} z v_f^z = z'_f \qquad (2.24)$$

Pada batasan (2.24) variabel $z'_f$ digunakan untuk melihat level ketinggian untuk setiap pesawat setelah optimasi.

### 2.2.4 Penggabungan Model VC dan AC

Penggabungan model mengakibatkan batasan (2.9 i) berubah menjadi

$$\delta_{ijz}^{1} + \delta_{ijz}^{2} + \delta_{ijz}^{3} + \delta_{ijz}^{4} + \delta_{ijz}^{5} = 1 \qquad (2.25)$$

Sehingga dalam variabel keputusan $\delta$ memiliki tiga kategori dengan dua model VC dimana pembagi dapat bernilai lebih besar dari 0 atau lebih kecil dari 0 dan model AC yang mengubah level ketinggian.

Fungsi obyektif pada model VC (2.1) dan model AC (2.16) akan berubah menjadi

$$\min w_1 \sum_{f \in F} \left( \frac{c_f^{q^+} q_f^+}{\bar{v}_f - \underline{v}_f} + \frac{c_f^{q^-} q_f^-}{\bar{v}_f - \underline{v}_f} \right) + w_2 \sum_{f \in F} c_f^j \rho_f \qquad (2.26)$$

variabel $w_n$ adalah beban yang di pertimbangkan untuk setiap fungsi obyektif. Nilai $w_1$ dan $w_2$ diset 0.5.

# BAB III
# PERANCANGAN PERANGKAT LUNAK

Bab ini akan menjelaskan perancangan model dari struktur pengambilan keputusan untuk menyelesaikan permasalahan penghindaran konflik menggunakan model VAC yang telah dijelaskan pada bab sebelumnya. Perancangan akan dibagi menjadi dua bagian yaitu perancangan data dan perancangan model VAC.

## 3.1 Model VAC Secara Umum

Penyelesaian permasalahan penghindaran konflik menggunakan struktur pengambilan keputusan model VAC terdapat satu program utama yang bertujuan untuk memberikan solusi terhadap permasalahan penghindaran konflik dengan menggunakan parameter-parameter dari hasil praproses dan data parameter yang melekat pada pesawat. Model VAC membutuhkan dua data masukan yaitu data masukan praproses dan data masukan parameter yang melekat pada pesawat dan menghasilkan data keluaran berupa kecepatan optimal dan ketinggian optimal yang memenuhi syarat tidak terjadinya konflik.

## 3.2 Perancangan Data

Perancangan data merupakan bagian yang penting dalam pengoperasian perangkat lunak karena dengan data yang benar, perangkat lunak dapat beroperasi dengan benar. Data yang diperlukan dalam pengoperasian perangkat lunak adalah data masukan (*input*), data proses yang dibutuhkan dan dihasilkan selama proses eksekusi perangkat lunak, dan data keluaran (*output*) yang memberikan hasil proses pengoperasian perangkat lunak untuk pengguna yang menggunakannya.

### 3.2.1 Data Masukan

Data masukan merupakan data yang dimasukkan oleh pengguna perangkat lunak sebagai *trigger* pada sistem penyelesaian masalah penghindaran konflik dengan menggunakan model VAC. Deta yang digunakan ada variabel yang melekat pada pesawat seperti id pesawat, koordinat(x,y), kecepatan, kecepatan maksimal, kecepatan minimal, *safety radius*, sudut arah terbang, dan level ketinggian. Data masukan ini akan digunakan pada saat praproses dan pada model VC Contoh data masukan dapat dilihat pada Tabel 3.1 dan Tabel 3.2.

**Tabel 3.1 Contoh Data Masukan(Bagian 1)**

| Pesawat | Level Ketinggian | Koordinat X | Koordinat Y | Sudut Arah Terbang | Kecepatan | Kecepatan Maksimal | Kecepatan Minimal | Safety radius |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | -900 | 900 | -1.5708 | 9 | 10 | 8 | 2.5 |
| 2 | 1 | -800 | 800 | 3.141593 | 9 | 10 | 8 | 2.5 |
| 3 | 1 | 250 | 750 | 0 | 9 | 10 | 8 | 2.5 |
| 4 | 1 | 750 | 750 | 3.141593 | 9 | 10 | 8 | 2.5 |
| 5 | 1 | -700 | 700 | 3.141593 | 9 | 10 | 8 | 2.5 |
| 6 | 1 | 150 | 600 | 2.356194 | 9 | 10 | 8 | 2.5 |
| 7 | 1 | 350 | 600 | 0.785398 | 9 | 10 | 8 | 2.5 |
| 8 | 3 | -900 | 500 | 0 | 9 | 10 | 8 | 2.5 |
| 9 | 3 | -500 | 500 | 0 | 9 | 10 | 8 | 2.5 |
| 10 | 3 | -350 | 500 | 3.141593 | 9 | 10 | 8 | 2.5 |
| 11 | 3 | 0 | 500 | -1.5708 | 9 | 10 | 8 | 2.5 |
| 12 | 1 | 500 | 500 | 1.570796 | 9 | 10 | 8 | 2.5 |
| 13 | 3 | 0 | 0 | 1.570796 | 9 | 10 | 8 | 2.5 |
| 14 | 4 | 500 | -100 | -1.5708 | 9 | 10 | 8 | 2.5 |
| 15 | 4 | 0 | -200 | 0 | 9 | 10 | 8 | 2.5 |
| 16 | 4 | 400 | -200 | 0 | 9 | 10 | 8 | 2.5 |

**Tabel 3.2 Contoh Data Masukan (Bagian 2)**

| Pesawat | Level Ketinggian | Koordinat X | Koordinat Y | Sudut Arah Terbang | Kecepatan | Kecepatan Maksimal | Kecepatan Minimal | Safety radius |
|---|---|---|---|---|---|---|---|---|
| 17 | 4 | 800 | -200 | 3.1415927 | 9 | 10 | 8 | 2.5 |
| 18 | 4 | 1000 | -200 | 3.1415927 | 100 | 101 | 99 | 2.5 |
| 19 | 4 | -700 | -300 | 0 | 9 | 10 | 8 | 2.5 |
| 20 | 2 | -700 | -300 | 0 | 9 | 10 | 8 | 2.5 |
| 21 | 2 | -300 | -300 | 3.1415927 | 9 | 10 | 8 | 2.5 |
| 22 | 3 | -300 | -300 | 3.1415927 | 9 | 10 | 8 | 2.5 |
| 23 | 3 | 0 | -300 | 1.5707963 | 100 | 101 | 99 | 2.5 |
| 24 | 4 | 700 | -300 | 3.1415927 | 9 | 10 | 8 | 2.5 |
| 25 | 2 | 0 | -400 | 2.3561945 | 9 | 10 | 8 | 2.5 |
| 26 | 2 | -500 | -500 | 1.5707963 | 9 | 10 | 8 | 2.5 |
| 27 | 2 | 0 | -500 | -2.356194 | 9 | 10 | 8 | 2.5 |
| 28 | 2 | 100 | -500 | -0.785398 | 9 | 10 | 8 | 2.5 |
| 29 | 3 | 250 | -500 | -0.785398 | 9 | 10 | 8 | 2.5 |
| 30 | 2 | -800 | -750 | 0 | 9 | 10 | 8 | 2.5 |
| 31 | 3 | 0 | -750 | 1.5707963 | 9 | 10 | 8 | 2.5 |
| 32 | 3 | 100 | -750 | 0 | 9 | 10 | 8 | 2.5 |
| 33 | 3 | 850 | -750 | 3.1415927 | 9 | 10 | 8 | 2.5 |
| 34 | 2 | -500 | -1000 | 0.7853982 | 9 | 10 | 8 | 2.5 |
| 35 | 2 | 0 | -1000 | 2.3561945 | 9 | 10 | 8 | 2.5 |
| 36 | 3 | 750 | -1000 | 2.3561945 | 9 | 10 | 8 | 2.5 |
| 37 | 3 | 900 | -1000 | 0.7853982 | 9 | 10 | 8 | 2.5 |

### 3.2.2 Data Keluaran

Data keluaran dari sistem ini yaitu data keluaran perubahan kecepatan, level pesawat setelah optimasi, dan banyaknya perubahan level ketinggian.

Contoh hasil keluaran berupa perubahan perubahan kecepatan, level pesawat setelah optimasi, dan banyaknya perubahan level ketinggian yang ditunjukkan pada Gambar 3.2 dan Gambar 3.3.

| [1] | q | q_plus | q_minus | rho | altitudeLevel |
|---|---|---|---|---|---|
| 1 | -0.61504 | 0 | 0.61504 | 0.000000 | 1 |
| 2 | 0.00000 | 0 | 0.00000 | 0.000000 | 1 |
| 3 | 0.00000 | 0 | 0.00000 | 1.000000 | 2 |
| 4 | 0.00000 | 0 | 0.00000 | 0.000000 | 1 |
| 5 | 0.00000 | 0 | 0.00000 | 0.000000 | 1 |
| 6 | 0.00000 | 0 | 0.00000 | 0.000000 | 1 |
| 7 | 0.00000 | 0 | 0.00000 | 0.000000 | 1 |
| 8 | 0.00000 | 0 | 0.00000 | 0.000000 | 3 |
| 9 | 0.00000 | 0 | 0.00000 | 0.000000 | 3 |
| 10 | 0.00000 | 0 | 0.00000 | 0.999999 | 2 |
| 11 | 0.00000 | 0 | 0.00000 | 0.999998 | 4 |
| 12 | -0.25103 | 0 | 0.25103 | 0.000000 | 1 |
| 13 | -0.12639 | 0 | 0.12639 | 0.000001 | 3 |
| 14 | -0.61504 | 0 | 0.61504 | 0.000000 | 4 |
| 15 | 0.00000 | 0 | 0.00000 | 0.000000 | 4 |
| 16 | 0.00000 | 0 | 0.00000 | 0.000000 | 4 |
| 17 | 0.00000 | 0 | 0.00000 | 2.000000 | 2 |
| 18 | 0.00000 | 0 | 0.00000 | 1.000000 | 3 |
| 19 | 0.00000 | 0 | 0.00000 | 0.000000 | 4 |
| 20 | 0.00000 | 0 | 0.00000 | 1.000000 | 1 |
| 21 | 0.00000 | 0 | 0.00000 | 0.000000 | 2 |
| 22 | 0.00000 | 0 | 0.00000 | 0.000000 | 3 |
| 23 | 0.00000 | 0 | 0.00000 | 0.999999 | 2 |
| 24 | 0.00000 | 0 | 0.00000 | 1.000000 | 3 |
| 25 | 0.00000 | 0 | 0.00000 | 0.000000 | 2 |
| 26 | -0.31272 | 0 | 0.31272 | 0.000000 | 2 |
| 27 | 0.00000 | 0 | 0.00000 | 0.999999 | 1 |
| 28 | 0.00000 | 0 | 0.00000 | 0.000000 | 2 |
| 29 | 0.00000 | 0 | 0.00000 | 0.000000 | 3 |
| 30 | 0.00000 | 0 | 0.00000 | 0.000000 | 2 |
| 31 | -0.12639 | 0 | 0.12639 | 0.000001 | 3 |
| 32 | 0.00000 | 0 | 0.00000 | 0.000000 | 3 |
| 33 | 0.00000 | 0 | 0.00000 | 1.000000 | 4 |
| 34 | -0.17823 | 0 | 0.17823 | 0.000000 | 2 |
| 35 | 0.00000 | 0 | 0.00000 | 0.000000 | 2 |
| 36 | 0.00000 | 0 | 0.00000 | 1.000000 | 2 |
| 37 | 0.00000 | 0 | 0.00000 | 0.000000 | 3 |

**Gambar 3.1 Hasil Keluaran Variasi Kecepatan, Perubahan Level dan Level Pesawat setelah Optimasi**

| | V_onthelevelZ | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 1 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 2 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 3 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 4 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 5 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 6 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 7 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 8 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 9 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 10 | 0.000000 | 0.999999 | 0.000000 | 0.000001 |
| 11 | 0.000000 | 0.000000 | 0.000002 | 0.999998 |
| 12 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 13 | 0.000000 | 0.000001 | 0.999998 | 0.000001 |
| 14 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 15 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 16 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 17 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 18 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 19 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 20 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| 21 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 22 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 23 | 0.000000 | 0.999999 | 0.000000 | 0.000001 |
| 24 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 25 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 26 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 27 | 0.999999 | 0.000000 | 0.000001 | 0.000000 |
| 28 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 29 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 30 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 31 | 0.000000 | 0.000001 | 0.999998 | 0.000001 |
| 32 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 33 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 34 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 35 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 36 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 37 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |

**Gambar 3.2 Level Pesawat setelah Optimasi**

Gambar 3.2 menunjukkan nilai *q* sebagai variabel perubahan kecepatan, *rho* sebagai variabel yang menyimpan banyaknya pe-rubahan level ketinggian, dan *altitudeLevel* adalah variabel yang bertujuan untuk melihat level ketinggian pesawat setelah optimasi. Gambar 3.3 menampilkan hasil level pesawat setelah optimasi yaitu *V_onthelevelZ*. *V_onthelevelZ* sama seperti *altitudeLevel* hanya ber-beda pada tipe data.

## 3.3 Perancangan dan *Pseudocode*

Pada bagian ini dijelaskan variabel-variabel yang digunakan mengenai rancangan model yang telah dijelaskan pada bab sebelumnya. Bagian ini terdiri atas variabel-variabel yang digunakan pada *pseudocode* praproses dan model VAC sebagai pengambil keputusan atas permasalahan penghindaran konflik.

### 3.3.1 Variabel yang Digunakan Pada Perancangan Penyelesaian Penghindaran Konflik Menggunakan Model VAC

Pada subbab ini dijelaskan bahwa variabel yang digunakan dalam *pseudocode* meliputi nama variabel, tipe, dan penjelasannya. Variabel yang terdapat pada *pseudocode* praproses penyelesaian penghindaran konflik menggunakan model ditunjukkan pada Tabel 3.3, 3.4, 3.5, dan 3.6, dan variabel untuk model utama penyelesaian penghindaran konflik dengan model VAC ditunjukkan pada dan Tabel 3.7, 3.8, 3.9, 3.10,dan 3.11.

**Tabel 3.3 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 1)**

| No | Nama Variabel | Tipe | Penjelasan |
|---|---|---|---|
| 1. | *koordinatXY* | *double* | Matriks untuk menyimpan data masukan koordinat x dan y |
| 2. | *KoordinatX* | *double* | Matriks untuk menyimpan data masukan koordinat X |
| 3. | *KoordinatY* | *double* | Matriks untuk menyimpan data masukan koordinat Y |
| 4. | *safetyRadius* | *double* | Matriks untuk menyimpan data masukan *safety radius* |
| 5. | *Motion* | *double* | Matriks untuk menyimpan data masukan *motion* atau sudut. |
| 6. | *Velocity* | *double* | Matriks untuk menyimpan data masukan *velocity* atau kecepatan. |
| 7. | *safetyDistant* | *double* | Matriks untuk menyimpan hasil penambahan *safety radius* antar pesawat |
| 8. | *aircraftDistant* | *double* | Matriks untuk menyimpan hasil perhitungan jarak antar pesawat |
| 9. | *sudutAlpha* | *double* | Matriks untuk menyimpan hasil perhitungan sudut α |
| 10. | *similarCoordinat* | *binary* | Matriks untuk menyimpan data pasangan pesawat yang termasuk dalam situasi *similar coordinates* |
| 11. | *SudutW* | *double* | Matriks untuk menyimpan hasil perhitungan sudut ω |
| 12. | *sudutWTop* | *double* | Matriks untuk menyimpan hasil perhitungan sudut $\widehat{\omega}$ |
| 13. | *SudutL* | *double* | Matriks untuk menyimpan hasil perhitungan sudut $l_{ij}$ |
| 14. | *SudutG* | *double* | Matriks untuk menyimpan hasil perhitungan sudut $G_{ij}$ |

**Tabel 3.4 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 2)**

| No | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 15. | *sudutLtop* | *double* | Matriks untuk menyimpan hasil perhitungan sudut $\hat{l}_{ij}$ |
| 16. | *sudutGtop* | *double* | Matriks untuk menyimpan hasil perhitungan sudut $\hat{g}_{ij}$ |
| 17. | *velocityVectorX* | *double* | Matriks untuk menyimpan hasil perhitungan *velocity vector* pada sumbu x |
| 18. | *velocityVectorY* | *double* | Matriks untuk menyimpan hasil perhitungan *velocity vector* pada sumbu y |
| 19. | *sudutWTops* | *double* | Matriks untuk menyimpan hasil perhitungan modifikasi sudut $\hat{\omega}$ |
| 20. | *intersectionPointInX* | *double* | Matriks untuk menyimpan hasil perhitungan intersection point atau titik potong pada koordinat x |
| 21. | *intersectionPointInY* | *double* | Matriks untuk menyimpan hasil perhitungan intersection point atau titik potong pada koordinat y |
| 22. | *headToHead* | *binary* | Matriks untuk menyimpan data pasangan pesawat yang termasuk dalam situasi head to head |

**Tabel 3.5 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 3)**

| No | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 23. | *altitudeConflict* | *binary* | Matriks untuk menyimpan data pasangan pesawat yang termasuk dalam situasi *similar coordinates* dan *head to head* |
| 24. | *distanceInterToAircrafti* | *double* | Matriks untuk menyimpan data jarak antara koordinat pesawat $f_i$ dan titik potong |
| 25. | *distanceInterToAircraftj* | *double* | Matriks untuk menyimpan data jarak antara koordinat pesawat $f_j$ dan titik potong |
| 26. | *distanceInterToMotioni* | *double* | Matriks untuk menyimpan data jarak antara koordinat pesawat $f_i$+ sudut dan titik potong |
| 27. | *distanceInterToMotionj* | *double* | Matriks untuk menyimpan data jarak antara koordinat pesawat $f_j$ + sudut dan titik potong |
| 28. | *subtractDistancei* | *double* | Matriks untuk menyimpan hasil pengurangan *d2ij-d1ij* |
| 29. | *subtractDistancej* | *double* | Matriks untuk menyimpan hasil pengurangan *d2ji-d1ji* |

**Tabel 3.6 Daftar Variabel yang Digunakan Pada Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 4)**

| No | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 30. | *falseConflict* | *binary* | Matriks untuk menyimpan data pasangan pesawat yang termasuk dalam situasi *false conflict* |
| 31. | *pathologicalCase* | *binary* | Matriks untuk menyimpan data pasangan pesawat yang termasuk dalam kasus anomali atau *pathological case* |
| 32. | *Hi* | *double* | Matriks untuk menyimpan hasil perhitungan $h_i$ |
| 33. | *Hj* | *double* | Matriks untuk menyimpan hasil perhitungan $h_j$ |
| 34. | *Ki* | *double* | Matriks untuk menyimpan hasil perhitungan $k_i$ |
| 35. | *Kj* | *double* | Matriks untuk menyimpan hasil perhitungan $k_j$ |
| 36. | *Hitop* | *double* | Matriks untuk menyimpan hasil perhitungan $h'_i$ |
| 37. | *Hjtop* | *double* | Matriks untuk menyimpan hasil perhitungan $h'_j$ |
| 38. | *Kitop* | *double* | Matriks untuk menyimpan hasil perhitungan $k'_i$ |
| 39. | *Kjtop* | *double* | Matriks untuk menyimpan hasil perhitungan $k'_j$ |

**Tabel 3.7 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 1)**

| No | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 1. | *Level* | *integer* | Matriks untuk menyimpan data masukan level ketinggian |
| 2. | *X* | *double* | Matriks untuk menyimpan data masukan koordinat X |
| 3. | *Y* | *double* | Matriks untuk menyimpan data masukan koordinat Y |
| 4. | *Sudut* | *double* | Matriks untuk menyimpan data masukan *motion* atau sudut |
| 5. | *v* | *double* | Matriks untuk menyimpan data masukan *velocity* atau kecepatan. |
| 6. | *v_max* | *double* | Matriks untuk menyimpan data masukan *maximum velocity* atau kecepatan maksimal |
| 7. | *v_min* | *double* | Matriks untuk menyimpan data masukan *minimum velocity* atau kecepatan minimal |
| 8. | *pathologicalCase* | *binary* | Matriks yang berasal dari praproses yang berisi data pasangan pesawat yang termasuk dalam kasus anomali atau *pathological case* |

**Tabel 3.8 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 2)**

| No | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 9. | *falseConflict* | *binary* | Matriks yang berasal dari praproses yang berisi data pasangan pesawat yang termasuk dalam situasi *false conflict* |
| 10. | *altitudeConflict* | *binary* | Matriks yang berasal dari praproses yang berisi data pasangan pesawat yang termasuk dalam situasi *similar coordinates* dan *head to head* |
| 11. | *Hi* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $h_i$ |
| 12. | *Hj* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $h_j$ |
| 13. | *Ki* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $k_i$ |
| 14. | *Kj* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $k_j$ |
| 15. | *Hitop* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $h'_i$ |
| 16. | *Hjtop* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $h'_j$ |

**Tabel 3.9 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 3)**

| No | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 17. | *Kitop* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $k'_i$ |
| 18. | *Kjtop* | *double* | Matriks yang berasal dari praproses yang berisi data hasil perhitungan $k'_j$ |
| 19. | *Q* | *double* | Variabel keputusan yang menyimpan variasi perubahan kecepatan untuk setiap pesawat |
| 20. | *q_plus* | *double* | Variabel keputusan yang menyimpan variasi perubahan kecepatan yang bernilai positif untuk setiap pesawat |
| 21. | *q_minus* | *double* | Variabel keputusan yang menyimpan variasi perubahan kecepatan yang bernilai negatif untuk setiap pesawat |
| 22. | *rho* | *integer* | Variabel keputusan yang menyimpan nilai perubahan level ketinggian untuk setiap pesawat |
| 23. | *V_onthelevelZ* | *binary* | Variabel keputusan yang bernilai 1 jika pesawat *f* berada pada level ketinggian *z* dan 0 jika sebaliknya |

**Tabel 3.10 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 4)**

| | | | |
|---|---|---|---|
| 24. | *delta_1* | *binary* | Variabel bantu terhadap model konstruksi geometri untuk setiap pasangan pesawat ($f_i$, $f_j$) dan setiap level ketinggian *z* |
| 25. | *delta_2* | *binary* | Variabel bantu terhadap model konstruksi geometri untuk setiap pasangan pesawat ($f_i$, $f_j$) dan setiap level ketinggian *z* |
| 26. | *delta_3* | *binary* | Variabel bantu terhadap model konstruksi geometri untuk setiap pasangan pesawat ($f_i$, $f_j$) dan setiap level ketinggian *z* |
| 27. | *delta_4* | *binary* | Variabel bantu terhadap model konstruksi geometri untuk setiap pasangan pesawat ($f_i$, $f_j$) dan setiap level ketinggian *z* |
| 28. | *delta_5* | *binary* | Variabel bantu terhadap model perubahan level untuk setiap pasangan pesawat ($f_i$, $f_j$) dan setiap level ketinggian *z* |
| 29. | *c_q_plus* | *integer* | Nilai biaya untuk perubahan kecepatan yang bernilai positif |
| 30. | *c_q_minus* | *integer* | Nilai biaya untuk perubahan kecepatan yang bernilai negatif |

**Tabel 3.11 Daftar Variabel yang Digunakan Pada Penyelesaian Penghindaran Konflik Menggunakan Model VAC (Bagian 5)**

| | | | |
|---|---|---|---|
| 31. | *c_j* | *integer* | Nilai biaya untuk perubahan level ketinggian |
| 32. | *w[n]* | *integer* | Nilai beban untuk setiap fungsi obyektif, *n* bernilai dari 1 hingga banyaknya fungsi obyektif. |
| 33. | Z | *integer* | Variabel yang berisi nilai banyaknya level ketinggian |
| 34. | *Aircraft* | *integer* | Variabel yang berisi nilai banyaknya pesawat |
| 35 | *altitudeLevel* | *Integer* | Variabel yang berisi level pesawat setelah optimasi |

### 3.3.2 Perancangan Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC

Berikut perancangan praproses pada pencarian solusi penghindaran konflik menggunakan model VAC yang ditunjukkan oleh sub-bab 3.3.2.1 sampai sub-bab 3.3.2.9. Perancangan praproses didefinisikan sebagai proses awal yang mendukung perancangan proses utama. Praproses menghasilkan variabel-variabel yang akan digunakan pada proses utama yaitu model VAC. *Pseudocode* akan dijelaskan berdasarkan alir model yang ditunjukkan oleh Gambar 2.1 dan sesuai dengan perumusan yang telah di jelaskan pada Bab II.

#### 3.3.2.1 Perancangan Perhitungan Sudut α dan *Similar Coordinates*

Pada perancangan sudut α dan *similar coordinates* juga akan dijelaskan perhitungan jarak antar pesawat dan perhitungan *safety distance*. Perhitungan jarak antar pesawat menggunakan

rumus *ecludian distance* dan perhitungan *safety distance* adalah penambahan *safety radius* antar pesawat. *Pseudocode* untuk jarak antar pesawat, *safety distance,* sudut α, dan *similar coordinates* ditunjukan pada Gambar 3.3.

| Masukan | Data koordinat x dan y(koordinatXY) dan data *safety radius* (safetyRadius) |
|---|---|
| Keluaran | jarak antar pesawat (aircraftDistant), safety distanance(safetyDistant), sudut α (sudutAlpha), dan *similar coordinates* (similarCoordinat) |

```
    // hitung jarak antar pesawat
1.  aircraftDistant = distance(koordinatXY);
    // hitung safety radius antar pesawat
2.  for i = 1 to row(safetyRadius);
3.       for j=1  to row(safetyRadius);
4.            safetyDistant[i,j] = safetyRadius[i]+
    safetyRadius[j];
5.       end;
6.  end;
    //hitung sudut alpha dan deteksi similar coordinates
7.  for i = 1 to row(safetyRadius)
8.        for j=1  to row(safetyRadius);
9.           if aircraftDistant[i,j]<=safetyDistant[i,j] then
10.          do;
11.              similarCoordinat[i,j]=1;
12.              sudutAlpha[i,j]=arsin(1);
13.          end;
14.          else
15.          do;
16.             similarCoordinat[i,j] =0;
17.             sudutAlpha[i,j]=
    arsin(safetyDistant[i,j]/aircraftDistant[i,j]);
18.              end;
19.      end;
20. end;
```

**Gambar 3.3** ***Pseudocode*** **Sudut α dan** ***Similar Coordinates***

Perancangan sudut α sesuai dengan formula (2.5 a) sub-bab 2.2.2.3 dan *similar coordinates* sesuai dengan sub-bab 2.2.3.4. Pada baris nomor 1 untuk menghitung jarak antar pesawat. Baris nomor 4 digunakan untuk menghitung *safety distance.* Kriteria pasangan

pesawat yang memiliki situasi *similar coordinates* dapat dilihat pada nomor 9, jika pasangan pesawat termasuk *similar coordinates* maka sudut α memiliki rumus *arcsin(1)*, ditunjukkan pada nomor 12. Jika tidak termasuk *similar coordinates,* sudut α ditujukan pada nomor 17 sesuai dengan formula (2.5 a). *similar coordinates* dirancang sesuai dengan formula (2.20).

### 3.3.2.2 Perancangan Perhitungan Sudut ω

Pada perancangan sudut ω, sudut ω dibagi menjadi 3 cara untuk koordinat x pada pesawat $f_i$ sama dengan koordinat x pesawat $f_j$ sudut ω ditetapkan bernilai 0 , koordinat y pada pesawat $f_i$ sama dengan koordinat y pesawat $f_j$ sudut ω ditetapkan bernilai π/2, dan untuk selain itu menggunakan formula yang telah dijelaskan pada Bab II (2.5 b). Perancangan menerapkan formula sesuai pada sub-bab 2.2.2.3. *Pseudocode* untuk sudut ω ditunjukkan pada Gambar 3.4.

| | |
|---|---|
| Masukan | Data koordinat x(koordinatX) dan data koordinat y(koordinatY) |
| Keluaran | sudut ω (sudutW) |

```
1.  for i = 1 to row(koordinatY)
2.      for j=1  to row(koordinatY)
3.        if  koordinatX[i,1]=koordinatX[j] | koordinatY[i] =
    koordinatY[j] then
4.           do;
5.              if  koordinatX[i]=koordinatX[j] then sudutW[i,j]=
    pi/2 /* kasus koordinat x sama */
6.              else sudutW[i,j]= 0; /* kasus koordinat y sama*/
7.           end;
8.              else sudutW[i,j]=atan((koordinatY[i] -
    koordinatY[j])/(koordinatX[i] - koordinatX[j])) /* kasus
    koordinat x dan y tidak sama*/
9.      end;
10. end;
```

**Gambar 3.4 *Pseudocode* Sudut ω**

Perancangan sudut ω sudah sesuai dengan pembahasan di sub-bab 2.2.2.3. Jika pesawat *i* memiliki koordinat x sama dengan

koordinat x pesawat *j* maka sudut ω bernilai π/2, hal ini ditunjukkan pada baris nomor 3 sebagai kriteria koordinat x yang sama dan baris nomor 5 sebagai hasil sudut ω untuk koordinat x yang sama. Jika pesawat *i* memiliki koordinat y sama dengan koordinat y pesawat *j* maka sudut ω bernilai 0, hal ini ditunjukkan pada baris nomor 3 sebagai kriteria koordinat y yang sama dan baris nomor 6 sebagai hasil sudut ω untuk koordinat y yang sama. Jika koordinat pesawat *i* dan pesawat *j,* sudut ω menggunakan rumusan (2.5 b) yang ditunjukkan pada baris nomor 8

### 3.3.2.3 Implementasi Perhitungan Sudut $\hat{\omega}$

Pada perancangan sudut $\hat{\omega}$, sudut $\hat{\omega}$ menggunakan sudut $\omega$ dan letak *quadrant* pesawat $f_j$ sesuai dengan pantauan dari pesawat $f_i$. Sudut $\hat{\omega}$ harus diantara(-π,π], untuk penjelasan sudut $\hat{\omega}$ dapat dilihat pada sub-bab 2.2.3.3. *Pseudocode* untuk sudut $\hat{\omega}$ ditunjukkan pada Gambar 3.5 dan 3.6.

| | |
|---|---|
| Masukan | Data koordinat x(koordinatX), data koordinat y(koordinatY), dan data sudut ω (sudutW) |
| Keluaran | sudut $\hat{\omega}$ (sudutWTop) dan modifikasi sudut $\hat{\omega}$ (sudutWTops) |

```
1.  for i = 1 to row(koordinatY);
2.      for j=1  to row(koordinatY);
           /*quadran I & IV */
3.          if koordinatX[j,]>koordinatX[i,] then sudutWTop[i,j]
    = sudutW[i,j];
         /*quadran II untuk Wtopij*/
4.         if koordinatX[j,]<=koordinatX[i,] &
    koordinatY[j,]>=koordinatY[i,]then sudutWTop[i,j] =
    sudutW[i,j] + pi;
        /*quadran III */
5.          if koordinatX[j,]<=koordinatX[i,] &
    koordinatY[j,]<koordinatY[i,]then sudutWTop[i,j] =
    sudutW[i,j] - pi;
6.      end;
7.  end;
8.  for i = 1 to row(koordinatY);
9.      for j=1  to row(koordinatY);
```

**Gambar 3.5 *Pseudocode* Sudut $\hat{\omega}$ (Bagian 1)**

```
            /*quadran I & IV*/
10.         if sudutWTop[i,j] = pi/2 + pi then sudutWTops[i,j] =
    pi/2;
11.         else sudutWTops[i,j] = sudutWTop[i,j];
12.     end;
13. end;
```

**Gambar 3.6 *Pseudocode* Sudut $\hat{\omega}$ (Bagian 2)**

Pada baris nomor 3 menunjukkan pesawat *j* berada di *quadrant* I atau IV dari pesawat *i*. Pada baris nomor 4 menunjukkan pesawat *j* berada di *quadrant* II dari pesawat *i*. Pada baris nomor 5 menunjukkan pesawat *j* berada di *quadrant* III dari pesawat *i*. Variabel *sudutWTops* digunakan untuk modifikasi variabel *sudutWTop*. Seperti diketahui sudut $\hat{\omega}$ harus diantara(-π,π] tetapi hasil sudutWtop menghasilkan (3π/2,π] sehingga harus dimodifikasi dan hasil modifikasi disimpan pada variabel *sudutWTops*. Kriteria sudut $\hat{\omega}$ yang memiliki nilai 3π/2 ditunjukkan pada baris nomor 10. *Pseudocode* $\hat{\omega}$ sesuai dengan sub-bab 2.2.3.3.

### 3.3.2.4 Perancangan Perhitungan Sudut $l, g, \hat{l}$, dan $\hat{g}$

Pada perancangan sudut $l, g, \hat{l}$, dan $\hat{g}$ menggunakan variabel *sudutL, sudutG, sudutLtop, dan sudutGtop*. Perancangan menerapkan formula yang tertera pada sub-bab 2.2.2.3 untuk sudut $l$ dan $g$ dan sub-bab 2.2.3.3 untuk sudut $\hat{l}$ dan $\hat{g}$. *Pseudocode* untuk sudut $l, g, \hat{l}$, dan $\hat{g}$ ditunjukkan pada Gambar 3.7.

| Masukan | data sudut ω (sudutW), sudut $\hat{\omega}$ (sudutWTops), dan sudut α (sudutAlpha) |
|---|---|
| Keluaran | sudut $l$ (sudutL), sudut $g$ (sudutG), sudut $\hat{l}$ (sudutLtop), dan sudut $\hat{g}$ (sudutGtop) |

```
1. sudutL = sudutW + sudutAlpha;
2. sudutG = sudutW - sudutAlpha;
3. sudutLtop = sudutWTops + sudutAlpha;
4. sudutGtop = sudutWTops - sudutAlpha;
```

**Gambar 3.7 *Pseudocode* Sudut $l, g, \hat{l}$, dan $\hat{g}$**

Pada baris nomor 1 menunjukkan rumusan sudut $l$ sesuai dengan formula (2.6 a), baris nomor 2 menunjukkan rumusan sudut $g$ sesuai dengan (2.6 b), baris nomor 3 menunjukkan rumusan sudut $\hat{l}$ sesuai dengan formula (2.19 b), dan baris nomor 4 menunjukkan rumusan sudut $\hat{g}$ sesuai dengan formula (2.19 a).

### 3.3.2.5 Perancangan Perhitungan Deteksi *Head to Head*

Pada perancangan deteksi situasi *head to head* menggunakan sudut $\widehat{\omega}$, sudut α, dan motion atau sudut arah terbang. *Pseudocode* untuk deteksi situasi *head to head* ditunjukkan pada Gambar 3.8.

| | |
|---|---|
| Masukan | sudut $\widehat{\omega}$ (sudutWTops), sudut α (sudutAlpha), dan sudut arah terbang(motion) |
| Keluaran | Deteksi *head to head* (headToHead) |

```
1.  for i = 1 to row(motion);
2.      for j=1  to row(motion);
3.           if sudutGtop[i,j]<=motion[i,] &
    motion[i,]<=sudutLtop[i,j] & sudutGtop[j,i]<=motion[j,] &
    motion[j,]<=sudutLtop[j,i] then headToHead[i,j]=1;
4.                else headToHead[i,j]=0;
5.      end;
6.  end;
```

**Gambar 3.8 *Pseudocode* Deteksi *Head to Head***

Pada baris nomor 3 menunjukkan kriteria yang termasuk situasi *head to head* yang dapat dilihat pada sub-bab 2.2.3.3 dengan formula (2.18).

### 3.3.2.6 Perancangan Perhitungan Titik Potong

Pada perancangan titik potong juga akan diikut sertakan implementasi *velocity vector* atau vector kecepatan. *Pseudocode* untuk perhitungan titik potong ditunjukkan pada Gambar 3.9

Perancangan diatas sesuai dengan formula titik potong yang dapat dilihat pada sub-bab 2.2.2.5.1. Pada baris nomor 3 menunjukkan *velocity vector* sumbu x dan baris nomor 4 menunjukkan *velo-*

*city vector* sumbu y. Perumusan *velocity vector* sudah sesuai dengan formula (2.3). Rumusan titik potong koordinat x dapat di-lihat pada baris nomor 9 dan untuk koordinat y dapat dilihat pada baris nomor 10, rancangan titik potong sudah sesaui dengan formula (2.15).

| | |
|---|---|
| Masukan | *Velocity* atau kecepatan (velocity), sudut arah terbang(motion), koordinat x (koordinatX), dan koordinat y (koordinatY) |
| Keluaran | Vector kecepatan (velocityVectorX,velocityVectorY) dan intersection point (intersectionPointInX, intersectionPointInY) |

```
1.  /* velocity vector (VectorX, VectorY)*/
2.  for i = 1 to row(safetyRadius);
3.       velocityVectorX[i]= velocity[i] * cosMotion[i];
4.       velocityVectorY[i]= velocity[i] * sinMotion[i];
5.  end;

6.  /* intersection point*/
7.  for i = 1 to row(motion);
8.       for j=1  to row(motion);
9.  intersectionPointInX[i,j] = (koordinatY[j,]-
    ((velocityVectorY[j,]/velocityVectorX[j,])*koordinatX[j,])-
    koordinatY[i,]+((velocityVectorY[i,]/velocityVectorX[i,])*koord
    inatX[i,]))/((velocityVectorY[i,]/velocityVectorX[i,])-
    (velocityVectorY[j,]/velocityVectorX[j,]);
10.             intersectionPointInY[i,j] =
    ((velocityVectorY[i,]/velocityVectorX[i,])*(koordinatY[j,]-
    ((velocityVectorY[j,]/velocityVectorX[j,])*koordinatX[j,]))-
    (velocityVectorY[j,]/velocityVectorX[j,])*(koordinatY[i,]-
    ((velocityVectorY[i,]/velocityVectorX[i,])*koordinatX[i,])))/((
    velocityVectorY[i,]/velocityVectorX[i,])-
    (velocityVectorY[j,]/velocityVectorX[j,]));
11.      end;
12. end;
```

**Gambar 3.9 *Pseudocode* Titik Potong**

### 3.3.2.7 Perancangan Perhitungan Deteksi Situasi *False Conflict*

Pada perancangan *false conflict* juga akan diikut sertakan perhitungan jarak posisi pesawat(x,y) dengan atau tanpa sudut ke tititk potong. *Pseudocode* untuk deteksi situasi *false conflict* ditun-jukkan pada Gambar 3.10.

| Masukan | intersection point (intersectionPointInX, intersectionPointInY), sudut arah terbang(motion), koordinat x (koordinatX), dan koordinat y (koordinatY) |
|---|---|
| Keluaran | Vector kecepatan (velocityVectorX,velocityVectorY) dan intersection point (intersectionPointInX, intersectionPointInY) |

```
    /* distance intersection point*/
1.  for i = 1 to row(motion);
2.      for j=1  to row(motion);
    /*jarak titik potong ke pesawat*/
3.              distanceInterToAircrafti[i,j] =
    sqrt(((koordinatX[i,]-
    intersectionPointInX[i,j])**2)+((koordinatY[i,]-
    intersectionPointInY[i,j])**2));
4.          distanceInterToAircraftj[i,j] = sqrt(((koordinatX[j,]-
    intersectionPointInX[i,j])**2)+((koordinatY[j,]-
    intersectionPointInY[i,j])**2));
    /*jarak titik potong ke pesawat dengan arah sudut*/
5.          distanceInterToMotioni[i,j] =
    sqrt((((koordinatX[i,]+cosMotion[i,])-
    intersectionPointInX[i,j])**2)+(((koordinatY[i,]+sinMotion[i,])
    -intersectionPointInY[i,j])**2));
6.  distanceInterToMotionj[i,j] =
    sqrt((((koordinatX[j,]+cosMotion[j,])-
    intersectionPointInX[i,j])**2)+(((koordinatY[j,]+sinMotion[j,])
    -intersectionPointInY[i,j])**2));
7.      end;
8.  end;
    /* penyederhanaan rumus kriteria false conflict*/
9.  subtractDistancei= distanceInterToMotioni-
    distanceInterToAircrafti;
10. subtractDistancej= distanceInterToMotionj-
    distanceInterToAircraftj;do i = 1 to nrow(motion);
    /*False conflict detection*/
11. for i = 1 to row(motion);
12.     for j=1  to row(motion);
13.              if subtractDistancei[i,j]>0 &
    subtractDistancej[i,j]>0 & altitudeConflict[i,j] ^=1 then
    falseConflict[i,j]= 1;
14.             else falseConflict[i,j] =0;
15.     end;
16. end;
```

**Gambar 3.10 *Pseudocode* Deteksi Situasi *False Conflict***

Perancangan diatas sesuai dengan formula *false conflict* pada sub-bab 2.2.2.5. Pada baris nomor 3 menunjukkan rumus jarak

antara titik potong dengan pesawat *i* yang merujuk pada formula (2.14 a) dan baris nomor 4 menunjukkan jarak antara titik potong dengan pesawat *j* yang merujuk pada formula (2.14 b). Pada baris nomor 5 menunjukkan rumus jarak antara titik potong dengan pesawat *i* ditambah sudut arah terbang yang merujuk pada formula (2.14 c) dan baris nomor 6 menunjukkan jarak antara titik potong dengan pesawat *j* ditambah sudut arah terbang yang merujuk pada formula (2.14 d). Pada baris nomor 9 menunjukkan pengurangan hasil jarak untuk pesawat *i* dan baris nomor 10 untuk pesawat *j*. kriteria *false conflict* ditunjukkan dengan baris nomor 13 sesuai dengan formula (2.13).

### 3.3.2.8 Perancangan Perhitungan Deteksi Kasus Anomali

Pada perancangan deteksi kasus anomali menggunakan koordinat x dan *safety distance*. *Pseudocode* untuk deteksi kasus anomali ditunjukkan pada Gambar 3.11. Pada baris nomor 3 menunjukkan kriteria yang termasuk kasus anomali yang dapat dilihat pada sub-bab 2.2.2.4. dengan formula (2.11).

| | |
|---|---|
| Masukan | Koordinat X (koordinatX) dan *safety distance* (safetyDisatant) |
| Keluaran | Deteksi kasus anomali(pathologicalCase) |

```
1.  for i = 1 to row(motion);
2.      for j=1  to row(motion);
3.          if abs(koordinatX[i,]-koordinatX[j,])<=
    safetyDistant[i,j] then pathologicalCase[i,j]=1;
4.              else pathologicalCase[i,j] =0;
5.      end; end;
6.  end;
```

**Gambar 3.11 *Pseudocode* Deteksi Kasus Anomali**

### 3.3.2.9 Perancangan Perhitungan Parameter Konstruksi Geometri

Pada perancangan parameter konstruksi geometri yang dimaksud adalah parameter $h_i, h_j, k_i, k_j$ dan $h'_i, h'_j, k'_i, k'_j$. *Pseudo-*

*code* untuk perhitungan parameter konstruksi geometri ditunjukkan pada Gambar 3.12

| Masukan | sudut $l$ (sudutL), sudut $g$ (sudutG), sudut $\hat{l}$ (sudutLtop), sudut $\hat{g}$ (sudutGtop), dan *motion* atau sudut arah terbang (motion) |
|---|---|
| Keluaran | $h_i$ (Hi), $h_j$ (Hj), $k_i$ (Ki), $k_j$ (Kj), $h'_i$ (Hitop), $h'_j$ (Hjtop), $k'_i$ (Kitop), dan $k'_j$ (Kjtop) |

```
1.  for i = 1 to row(motion);
2.      for j=1  to row(motion);
3.              Hi[i,j] = (tan(sudutL[i,j])*cos(motion[i]))-
    (sin(motion[i])) ;
4.              Hj[i,j] = (tan(sudutL[i,j])*cos(motion[j]))-
    (sin(motion[j])) ;
5.             Ki[i,j] = (tan(sudutG[i,j])*cos(motion[i]))-
    (sin(motion[i])) ;
6.              Kj[i,j] = (tan(sudutG[i,j])*cos(motion[j]))-
    (sin(motion[j])) ;
7.              Hitop[i,j] = (-tan(sudutL[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
8.              Hjtop[i,j] = (-tan(sudutL[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
9.          Kitop[i,j] = (-tan(sudutG[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
10.             Kjtop[i,j] = (-tan(sudutG[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
11.     end;
12. end;
```

**Gambar 3.12 *Pseudocode* Parameter Konstruksi Geometri**

Pada baris nomor 3 menunjukkan parameter konstruksi geometri $h_i$ yang merujuk pada formula (2.6 a), baris nomor 4 menunjukkan parameter konstruksi geometri $h_j$ yang merujuk pada formula (2.6 b), baris nomor 5 menunjukkan parameter konstruksi geometri $k_i$ yang merujuk pada formula (2.6 c), baris nomor 6 menunjukkan parameter konstruksi geometri $k_j$ yang merujuk pada formula (2.6 d). Rujukan formula $h_i, h_j, k_i$, dan $k_j$ dapat dilihat di sub-bab 2.2.2.3. Dan untuk baris nomor 7 menunjukkan parameter konstruksi geometri $h'_i$ yang merujuk pada formula (2.12 a), baris nomor 8 menunjukkan parameter konstruksi geometri $h'_j$ yang me-rujuk pada formula (2.12 b), baris nomor 9 menunjukkan parameter kons-

truksi geometri $k'_i$ yang merujuk pada formula (2.12 c), baris nomor 10 menunjukkan parameter konstruksi geometri $k'_j$ yang merujuk pada formula (2.12 d). Rujukan formula $h'_i, h'_j, k'_i,$ dan $k'_j$ dapat dilihat pada sub-bab 2.2.2.4.

### 3.3.3 Perancangan Model VAC untuk Penyelesaian Masalah Penghindaran Konflik

Berikut perancangan pencarian solusi penghindaran konflik menggunakan model VAC yang ditunjukkan oleh Gambar 3.13, 3.14, 3.15, 3.16, dan 3.17. Perancangan model utama ini bertujuan untuk mendapatkan kecepatan optimal dengan menambahkan kecepatan awal dengan variasi kecepatan dan menempatkan pesawat di level ketinggian yang tidak akan mengalami konflik dengan pesawat lainnya. Model utama menghasilkan variasi kecepatan *q*, level ketinggian setelah optimasi yang disimpan pada variabel *V_onthelevelZ* dan *altitudeLevel* serta jumlah perubahan level yang dialami setiap pesawat *rho*. *Pseudocode* akan dijelaskan sesuai dengan penjelasan pada Bab II.

| | |
|---|---|
| Masukan | $h_i$ (Hi), $h_j$ (Hj), $k_i$ (Ki), $k_j$ (Kj), $h'_i$ (Hitop), $h'_j$ (Hjtop), $k'_i$ (Kitop), $k'_j$ (Kjtop), Deteksi kasus anomali(pathologicalCase), deteksi false conflict (falseConflict), deteksi SC dan HTH (altitudeConflict), *motion* atau sudut arah terbang (sudut), koordinat x (x), koordinat y (y), *velocity*/kecepatan (v), kecepatan maksimal (v_max), kecepatan minimal (v_min),level ketinggian pesawat (level), nilai biaya perubahan kecepatan positif (c_q_plus), nilai biaya perubahan kecepatan negatif (c_q_minus), nilai biaya perubahan ketinggian(c_j), banyaknya level ketinggian (z), nilai beban fungsi obyektif VC ($w_1$) , nilai beban fungsi obyektif VC ($w_2$), banyaknya pesawat (Aircraft) |
| Keluaran | Variasi kecepatan (q), variasi kecepatan positif (q_plus), variasi kecepatan negatif (q_minus), banyaknya perubahan level ketinggian (rho), level ketinggian setelah optimasi (V_onthelevelZ, altitudeLevel), variabel pemilihan model (delta_1, delta 2, delta 3, delta 4, delta 5) |
| **1.** Set w[1] = 0.5 | |

**Gambar 3.13 *Pseudocode* Model VAC (Bagian 1)**

```
2.  Set w[2] = 0.5
3.  Set c_q_plus = 1
4.  Set c_q_minus = 1
5.  Set c_j = 1
6.  Set number z = 1..4

7.  Read Aircraft
8.  Read Hi
9.  Read Hj
10. Read Ki
11. Read Kj
12. Read Hitop
13. Read Hjtop
14. Read Kitop
15. Read Kjtop
16. Read pathologicalCase
17. Read falseConflict
18. Read altitudeConflict
19. Read sudut
20. Read x
21. Read y
22. Read v
23. Read v_max
24. Read v_min
25. Read level
26. VARIABEL q{Aircraft}
27. VARIABEL q_plus{Aircraft}>=0
28. VARIABEL q_minus{Aircraft}>=0
29. VARIABEL alpha{Aircraft} BINARY
30. VARIABEL b{Aircraft} BINARY
31. VARIABEL rho{Aircraft} INTEGER >=0
32. VARIABEL beta{Aircraft}>=0
33. VARIABEL V_onthelevelZ{Aircraft,z} BINARY
34. VARIABEL delta_1 {i,j,z} BINARY
35. VARIABEL delta_2 {i,j,z} BINARY
36. VARIABEL delta_3 {i,j,z} BINARY
37. VARIABEL delta_4 {i,j,z} BINARY
38. VARIABEL delta_5 {i,j,z} BINARY
39. VARIABEL altitudeLevel{Aircraft} INTEGER >=0
```

**Gambar 3.14 *Pseudocode* Model VAC (Bagian 2)**

```
/*DECLARE OBJECTIVE*/
40. MIN VAC_model = w [1] * ( sum { f in Aircraft} ( ( (c_q_plus[f]
* q_plus[f])/ ( v_max [f] -  v_min [f] ) ) + (c_q_minus[f] *
q_minus[f])/ ( v_max [f] -  v_min [f] ) )) + w[2] * ( sum { f
in Aircraft} ( c_j[f]* rho[f]));

/* CONSTRAINT (10) */
41. CON CONSTRAINT_10 {f in Aircraft} :
v_min[f] <= ( v[f] + q[f] ) <= v_max[f];

/* CONSTRAINT (11) */
42. CON CONSTRAINT_11 { <f1,f2>  in {i,j}, le in z : f1 < f2 &
falseConflict[f1,f2]=0} :
((v[f1] + q[f1]) * ((cos(sudut [f1]) * (1 -
pathologicalCase[f1,f2])) - (sin(sudut[f1]) *
pathologicalCase[f1,f2])))-(( v[f2] + q[f2] ) * ((
cos(sudut[f2]) * ( 1 - pathologicalCase[f1,f2])) -
(sin(sudut[f2]) * pathologicalCase[f1,f2] )))  <= (( v_max [f1]
+ v_max [f2] ) * (1 - delta_1 [f1,f2,le]));

/* CONSTRAINT (12) */
43. CON CONSTRAINT_12  { <f1,f2>  in {i,j}, le in z : f1 < f2 &
falseConflict[f1,f2]=0}:
 - ((v[f1] + q[f1]) *((Hi[f1,f2]* ( 1 -
pathologicalCase[f1,f2])) + (Hitop[f1,f2] *
pathologicalCase[f1,f2])))  + ((v[f2] + q[f2]) *((Hj[f1,f2]* (
1 - pathologicalCase[f1,f2])) + (Hjtop[f1,f2] *
pathologicalCase[f1,f2]))) <= (((((v_max [f1] * abs(Hi[f1,f2]))
+ ( v_max [f2] * abs(Hj[f1,f2]) )) * ( 1 -
pathologicalCase[f1,f2])) + (((v_max [f1] * abs(Hitop [f1,f2]))
+ ( v_max [f2] * abs(Hjtop [f1,f2]))) *
pathologicalCase[f1,f2])) * (1 - delta_1 [f1,f2,le]));

/* CONSTRAINT (13) */
44. CON CONSTRAINT_13 { <f1,f2>  in {i,j}, le in z : f1 < f2  &
falseConflict[f1,f2]=0}:
 ((v[f1] + q[f1]) * ((cos(sudut[f1]) * (1 -
pathologicalCase[f1,f2])) - (sin(sudut[f1]) *
pathologicalCase[f1,f2])))-(( v[f2] + q[f2] ) * (( cos(sudut
[f2]) * ( 1 - pathologicalCase[f1,f2])) - (sin(sudut[f2]) *
pathologicalCase[f1,f2] ))) <= (( v_max [f1] + v_max [f2] ) *
(1 - delta_2 [f1,f2,le]));

/* CONSTRAINT (14) */
45. CON CONSTRAINT_14 { <f1,f2>  in {i,j}, le in z: f1 < f2  &
falseConflict[f1,f2]=0}:
((v[f1] + q[f1]) *((Ki[f1,f2]* ( 1 - pathologicalCase[f1,f2]))
+ (Kitop[f1,f2] * pathologicalCase[f1,f2])))  - ((v[f2] +
q[f2]) *((Kj[f1,f2]* ( 1 - pathologicalCase[f1,f2])) +
(Kjtop[f1,f2] * pathologicalCase[f1,f2]))) <= (((((v_max [f1] *
abs(Ki[f1,f2])) + ( v_max [f2] * abs(Kj[f1,f2]) )) * ( 1 -
pathologicalCase[f1,f2])) + (((v_max [f1] * abs(Kitop [f1,f2]))
+ ( v_max [f2] * abs(Kjtop [f1,f2]))) *
pathologicalCase[f1,f2])) * (1 - delta_2 [f1,f2,le]));
```

**Gambar 3.15 *Pseudocode* Model VAC (Bagian 3)**

```
    /* CONSTRAINT (15) */
46. CON CONSTRAINT_15 { <f1,f2>  in {i,j}, le in z: f1 < f2  &
    falseConflict[f1,f2]=0}:
    -((v[f1] + q[f1]) * ((cos(sudut [f1]) * (1 -
    pathologicalCase[f1,f2])) - (sin(sudut [f1]) *
    pathologicalCase[f1,f2]))) +(( v[f2] + q[f2] ) * (( cos(sudut
    [f2]) * ( 1 - pathologicalCase[f1,f2])) - (sin(sudut[f2]) *
    pathologicalCase[f1,f2] )))  <= (( v_max [f1] + v_max [f2] ) *
    (1 - delta_3 [f1,f2,le]));

    /* CONSTRAINT (16) */
47. CON CONSTRAINT_16 { <f1,f2>  in {i,j}, le in z: f1 < f2  &
    falseConflict[f1,f2]=0}:
     ((v[f1] + q[f1]) *((Hi[f1,f2]* ( 1 - pathologicalCase[f1,f2]))
    + (Hitop[f1,f2] * pathologicalCase[f1,f2])))  - ((v[f2] +
    q[f2]) *((Hj[f1,f2]* ( 1 - pathologicalCase[f1,f2])) +
    (Hjtop[f1,f2] * pathologicalCase[f1,f2]))) <= (((((v_max [f1] *
    abs(Hi[f1,f2])) + ( v_max [f2] * abs(Hj[f1,f2]) )) * ( 1 -
    pathologicalCase[f1,f2])) + (((v_max [f1] * abs(Hitop [f1,f2]))
    + ( v_max [f2] * abs(Hjtop [f1,f2]))) *
    pathologicalCase[f1,f2])) * (1 - delta_3 [f1,f2,le]));

    /* CONSTRAINT (17) */
48. CON CONSTRAINT_17  { <f1,f2>  in {i,j}, le in z : f1 < f2  &
    falseConflict[f1,f2]=0}:
    -((v[f1] + q[f1]) * ((cos(sudut [f1]) * (1 -
    pathologicalCase[f1,f2])) - (sin(sudut [f1]) *
    pathologicalCase[f1,f2]))) +(( v[f2] + q[f2] ) * (( cos(sudut
    [f2]) * ( 1 - pathologicalCase[f1,f2])) - (sin(sudut[f2]) *
    pathologicalCase[f1,f2] ))) <= (( v_max [f1] + v_max [f2] ) *
    (1 - delta_4 [f1,f2,le]));

    /* CONSTRAINT (18) */
49. CON CONSTRAINT_18  { <f1,f2>  in {i,j}, le in z: f1 < f2 &
    falseConflict[f1,f2]=0}:
    - ((v[f1] + q[f1]) *((Ki[f1,f2]* ( 1 -
    pathologicalCase[f1,f2])) + (Kitop[f1,f2] *
    pathologicalCase[f1,f2])))  + ((v[f2] + q[f2]) *((Kj[f1,f2]* (
    1 - pathologicalCase[f1,f2])) + (Kjtop[f1,f2] *
    pathologicalCase[f1,f2]))) <= (((((v_max [f1] * abs(Ki[f1,f2]))
    + ( v_max [f2] * abs(Kj[f1,f2]) )) * ( 1 -
    pathologicalCase[f1,f2])) + (((v_max [f1] * abs(Kitop [f1,f2]))
    + ( v_max [f2] * abs(Kjtop [f1,f2]))) *
    pathologicalCase[f1,f2])) * (1 - delta_4 [f1,f2,le]));

    /* CONSTRAINT (19) */
50. CON CONSTRAINT_19 { <f1,f2>  in {i,j}, le in z: f1 < f2 &
    falseConflict[f1,f2]=0 }:
    delta_1 [f1,f2,le] + delta_2 [f1,f2,le] + delta_3 [f1,f2,le] +
    delta_4 [f1,f2,le] + delta_5 [f1,f2,le] = 1;
```

**Gambar 3.16 *Pseudocode* Model VAC (Bagian 4)**

```
    /* CONSTRAINT (20) */
51. CON CONSTRAINT_20 { <f1,f2>  in {i,j}, le in z : f1 < f2 }:
    altitudeConflict[f1,f2] <= altitudeConflict[f1,f2] * delta_5
    [f1,f2,le] ;

    /* CONSTRAINT (21) */
52. CON CONSTRAINT_21 { <f1,f2>  in {i,j}, le in z : f1 < f2}:
    V_onthelevelZ [f1,le] + V_onthelevelZ [f2,le] >= delta_5
    [f1,f2,le] - 1 ;

    /* CONSTRAINT (22) */
53. CON CONSTRAINT_22 { <f1,f2>  in {i,j}, le in z : f1 < f2}:
    V_onthelevelZ [f1,le] + V_onthelevelZ [f2,le] <= 2 - delta_5
    [f1,f2,le] ;

    /* CONSTRAINT (23) */
54. CON CONSTRAINT_23 {f in Aircraft}:
    sum { le in z} V_onthelevelZ [f,le] = 1 ;

    /* CONSTRAINT (27) */
55. CON CONSTRAINT_27 {f in Aircraft}:
    (sum { le in z} ( le* V_onthelevelZ[f, le])) - level[f] <=
    rho[f];

    /* CONSTRAINT (28) */
56. CON CONSTRAINT_28 {f in Aircraft} :
    level[f] - (sum { le in z} ( le* V_onthelevelZ[f,le])) <=
    rho[f];

57. CON CONSTRAINT_33_2 {f in Aircraft } :
    q[f] = q_plus[f] - q_minus[f];

58. CON CONSTRAINT_33{f in Aircraft } :
    (sum { le in z} ( le* V_onthelevelZ[f, le]))= altitudeLevel[f];

59. SOLVE;
    print q q_plus q_minus rho  altitudeLevel;
```

**Gambar 3.17 *Pseudocode* Model VAC (Bagian 5)**

Pada baris nomor 40 menunjukkan fungsi obyektif model VAC yang sesuai dengan sub-bab 2.2.4 dengan formula (2.26). Pada baris nomor 42 sampai 50 menunjukkan batasan konstruksi geometri pada sub-bab 2.2.2.3 nomor (2.9) dengan modifikasi model dengan kasus anomali yang dibahasa pada sub-bab 2.2.2.4. Baris nomor 50 sesuai dengan formula batasan (2.25) pada sub-bab 2.2.4. Batasan perbedaan ketinggian model AC sub-bab 2.2.3.5 ditunjuk-

kan pada model VAC di nomor 51 sampai nomor 53. Pada baris nomor 51 menunjukkan batasan dengan formula (2.21) dimana $hth_{ij} + sc_{ij}$ sudah digabung menjadi satu parameter yaitu *altitudeConflict*. Penggabungan $hth_{ij} + sc_{ij}$ dilakukan pada saat praproses. Pada baris nomor 52 menunjukkan batasan dengan formula (2.22 a) dan untuk baris nomor 53 menunjukkan formula (2.22 b). Batasan ketinggian sub-bab 2.2.3.2 dengan formula (2.17) ditunjukkan pada nomor 54. Batasan perubahan level yang dijelaskan pada sub-bab 2.2.3.6 ditunjukkan pada baris nomor 55, 56, dan 58. Pada nomor 55 menujukkan batasan (2.23 a) dan nomor 56 menunjukkan batasan (2.23 b). Nomor 58 menunjukkan batasan (2.24). Batasan variasi kecepatan dapat dilihat pada baris nomor 41 untuk formula (2.2 a) dan baris nomor 57 untuk formula (2.2 b).

# BAB IV
# IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi yang meliputi algoritma dan kode sumber yang digunakan untuk melakukan proses penyelesaian permasalahan penghindaran konflik dengan menggunakan model VAC menggunakan SAS.

## 4.1 Lingkungan Implementasi

Implementasi penyelesaian permasalahan penghindaran konflik menggunakan model VAC menggunakan spesifikasi perangkat keras dan perangkat lunak yang ditunjukkan pada Tabel 4.1.

**Tabel 4.1 Lingkungan Implementasi Perangkat Lunak**

| | |
|---|---|
| Perangkat Keras | Prosesor: Intel(R) Core(TM) i7 – 2350 MCPU @ 2.30 GHz<br>Memori: 4 GB |
| Perangkat Lunak | Sistem Operasi*:* Microsoft Windows 8 Pro 32-*bit*<br>Perangkat Pengembang*:* SAS 9.2 |

## 4.2 Implementasi

Implementasi sistem perangkat lunak penyelesaian permasalahan penghindaran konflik dengan menggunakan model VAC dilakukan sesuai dengan proses yang dijelaskan pada bab sebelumnya. Program utama penghindaran konflik adalah berisi batasan batasan dan fungsi obyektif dari model VAC dan program pendukung berisi praproses model VAC, hasil praproses mendapatkan beberapa parameter yang akan digunakan pada program utama. Setiap bagian dijabarkan lebih lanjut pada sub-bab berikutnya.

### 4.2.1 Implementasi Praproses Penyelesaian Penghindaran Konflik Menggunakan Model VAC

Berikut Implementasi praproses pada pencarian solusi penghindaran konflik menggunakan model VAC yang ditunjukkan oleh sub-bab 4.2.1.1 sampai sub-bab 4.2.1.9. Implementasi praproses adalah proses awal yang mendukung implementasi proses utama. Praproses menghasilkan variabel-variabel yang akan digunakan pada proses utama yaitu model VAC. Implementasi akan dijelaskan berdasarkan *pseudocode* yang telah dijelaskan pada Bab III.

#### 4.2.1.1 Implementasi Perhitungan Sudut α dan *Similar Coordinates*

Pada implementasi sudut α dan *similar coordinates* disesuaikan dengan rancangan sudut α dan *similar coordinates* pada sub-bab 3.3.2.1. Implementasi untuk jarak antar pesawat, *safety distance,* sudut α, dan *similar coordinates* ditunjukkan pada Kode Sumber 4.1 dan Kode Sumber 4.2.

| | |
|---|---|
| Masukan | Data koordinat x dan y(koordinatXY) dan data *safety radius* (safetyRadius) |
| Keluaran | jarak antar pesawat (aircraftDistant), safety distanance(safetyDistant), sudut α (sudutAlpha), dan *similar coordinates* (similarCoordinat) |

```
1.  aircraftDistant = distance(koordinatXY);
2.  do i = 1 to nrow(safetyRadius);
3.       do j=1  to nrow(safetyRadius);
4.            safetyDistant[i,j] = safetyRadius[i]+
    safetyRadius[j];
5.       end;
6.  end;
7.  do i = 1 to nrow(safetyRadius)
8.        do j=1  to nrow(safetyRadius);
```

**Kode Sumber 4.1 Implementasi Sudut α dan Similar Coordinates (Bagian 1)**

```
9.          if aircraftDistant[i,j]<=safetyDistant[i,j] then
10.         do;
11.             similarCoordinat[i,j]=1;
12.             sudutAlpha[i,j]=arsin(1);
13.         end;
14.         else
15.         do;
16.            similarCoordinat[i,j] =0;
17.            sudutAlpha[i,j]=
    arsin(safetyDistant[i,j]/aircraftDistant[i,j]);
18.             end;
19.     end;
20. end;
```

**Kode Sumber 4.2 Implementasi Sudut α dan *Similar Coordinates* (Bagian 2)**

### 4.2.1.2 Implementasi Perhitungan Sudut ω

Pada implementasi sudut ω disesuaikan dengan rancangan sudut ω pada sub-bab 3.3.2.2. Implementasi untuk sudut ω ditunjukkan pada Kode Sumber 4. 3.

| Masukan | Data koordinat x(koordinatX) dan data koordinat y(koordinatY) |
|---|---|
| Keluaran | sudut ω (sudutW) |

```
1.  do i = 1 to row(koordinatY)
2.      do j=1  to row(koordinatY)
3.          if  koordinatX[i,1]=koordinatX[j] | koordinatY[i] =
    koordinatY[j] then
4.              do;
5.                 if  koordinatX[i]=koordinatX[j] then
    sudutW[i,j]= pi/2
6.                 else sudutW[i,j]= 0;
7.              end;
8.              else sudutW[i,j]=atan((koordinatY[i] -
    koordinatY[j])/(koordinatX[i] - koordinatX[j]))
9.      end;
10. end;
```

**Kode Sumber 4.3 *Implementasi* Sudut ω**

#### 4.2.1.3 Implementasi Perhitungan Sudut $\hat{\omega}$

Pada implementasi sudut $\hat{\omega}$, sudut $\hat{\omega}$ menggunakan sudut $\omega$ dan letak *quadrant* pesawat $f_j$ sesuai dengan pantauan dari pesawat $f_i$ . Sudut $\hat{\omega}$ diantara(-π,π], untuk penjelasan sudut $\hat{\omega}$ dapat dilihat pada sub-bab 2.2.3.3. Implementasi disesuaikan dengan perancangan sudut $\hat{\omega}$ pada sub-bab 3.3.2.3 Implementasi untuk sudut $\hat{\omega}$ ditunjukkan pada Kode Sumber 4.4.

| Masukan | Data koordinat x(koordinatX), data koordinat y(koordinatY), dan data sudut ω (sudutW) |
|---|---|
| Keluaran | sudut $\hat{\omega}$ (sudutWTop) dan modifikasi sudut $\hat{\omega}$ (sudutWTops) |

```
1.  do i = 1 to row(koordinatY);
2.      do j=1  to row(koordinatY);
           /*quadran I & IV */
3.          if koordinatX[j,]>koordinatX[i,] then
    sudutWTop[i,j] = sudutW[i,j];
          /*quadran II untuk Wtopij*/
4.          if koordinatX[j,]<=koordinatX[i,] &
    koordinatY[j,]>=koordinatY[i,]then sudutWTop[i,j] =
    sudutW[i,j] + pi;
          /*quadran III */
5.          if koordinatX[j,]<=koordinatX[i,] &
    koordinatY[j,]<koordinatY[i,]then sudutWTop[i,j] =
    sudutW[i,j] - pi;
6.      end;
7.  end;
8.  do i = 1 to row(koordinatY);
9.      do j=1  to row(koordinatY);
           /*quadran I & IV*/
10.         if sudutWTop[i,j] = pi/2 + pi then
    sudutWTops[i,j] = pi/2;
11.         else sudutWTops[i,j] = sudutWTop[i,j];
12.     end;
13. end;
14. print sudutWTops;
```

**Kode Sumber 4.4 Implementasi Sudut $\hat{\omega}$**

#### 4.2.1.4 Implementasi Perhitungan Sudut $l, g, \hat{l}$, dan $\hat{g}$

Pada implementasi sudut $l, g, \hat{l}$, dan $\hat{g}$ menggunakan variabel *sudutL, sudutG, sudutLtop*, dan *sudutGtop*. Implementasi menerapkan formula yang tertera pada sub-bab 2.2..2.3 dan implementasi disesuaikan dengan perancangan sudut $l, g, \hat{l}$, dan $\hat{g}$ pada sub-bab 3.3.2.4. Implementasi untuk sudut $l, g, \hat{l}$, dan $\hat{g}$ ditunjukkan pada Kode Sumber 4.5.

| Masukan | data sudut ω (sudutW), sudut $\hat{\omega}$ (sudutWTops), dan sudut α (sudutAlpha) |
|---|---|
| Keluaran | sudut $l$ (sudutL), sudut $g$ (sudutG), sudut $\hat{l}$ (sudutLtop), dan sudut $\hat{g}$ (sudutGtop) |

```
1. sudutL = sudutW + sudutAlpha;
2. sudutG = sudutW - sudutAlpha;
3. sudutLtop = sudutWTops + sudutAlpha;
4. sudutGtop = sudutWTops - sudutAlpha;
```

**Kode Sumber 4.5 Implementasi Sudut $l, g, \hat{l}$, dan $\hat{g}$**

#### 4.2.1.5 Implementasi Perhitungan Deteksi *Head to Head*

Pada implementasi deteksi situasi *head to head* menggunakan sudut $\hat{\omega}$, sudut α, dan motion atau sudut arah terbang. Implementasi disesuaikan dengan perancangan deteksi situasi *head to head* pada sub-bab 3.3.2.5 Implementasi untuk deteksi situasi *head to head* ditunjukkan pada Kode Sumber 4.6.

| Masukan | sudut $\hat{\omega}$ (sudutWTops), sudut α (sudutAlpha), dan sudut arah terbang(motion) |
|---|---|
| Keluaran | Deteksi *head to head* (headToHead) |

```
1. do i = 1 to row(motion);
2.      do j=1  to row(motion);
3.           if sudutGtop[i,j]<=motion[i,] &
   motion[i,]<=sudutLtop[i,j] & sudutGtop[j,i]<=motion[j,] &
   motion[j,]<=sudutLtop[j,i] then headToHead[i,j]=1;
4.                else headToHead[i,j]=0;
5.      end; end;
```

**Kode Sumber 4.6 Implementasi Deteksi *Head to Head***

#### 4.2.1.6 Implementasi Perhitungan Titik Potong

Pada implementasi titik potong juga akan diikut sertakan Implementasi *velocity vector* atau vector kecepatan. Implementasi diatas sesuai dengan formula titik potong yang dapat dilihat pada sub-bab 2.2.2.5.1 dan perancangan titik potong pada sub-bab 3.3.2.6. Implementasi untuk perhitungan titik potong ditunjukkan pada Kode Sumber 4.7.

| | |
|---|---|
| Masukan | *Velocity*/kecepatan (velocity), sudut arah terbang(motion), koordinat x (koordinatX), dan koordinat y (koordinatY) |
| Keluaran | Vector kecepatan (velocityVectorX,velocityVectorY) dan intersection point (intersectionPointInX, intersectionPointInY) |

```
1.  /* velocity vector (VectorX, VectorY)*/
2.  do i = 1 to nrow(safetyRadius);
3.       velocityVectorX[i]= velocity[i] * cosMotion[i];
4.       velocityVectorY[i]= velocity[i] * sinMotion[i];
5.  end;
6.  /* intersection point*/
7.  do i = 1 to nrow(motion);
8.       do j=1  to nrow(motion);
9.  intersectionPointInX[i,j] = (koordinatY[j,]-
    ((velocityVectorY[j,]/velocityVectorX[j,])*koordinatX[j,])-
    koordinatY[i,]+((velocityVectorY[i,]/velocityVectorX[i,])*koord
    inatX[i,]))/((velocityVectorY[i,]/velocityVectorX[i,])-
    (velocityVectorY[j,]/velocityVectorX[j,]);
10. intersectionPointInY[i,j] =
    ((velocityVectorY[i,]/velocityVectorX[i,])*(koordinatY[j,]-
    ((velocityVectorY[j,]/velocityVectorX[j,])*koordinatX[j,]))-
    (velocityVectorY[j,]/velocityVectorX[j,])*(koordinatY[i,]-
    ((velocityVectorY[i,]/velocityVectorX[i,])*koordinatX[i,])))/((
    velocityVectorY[i,]/velocityVectorX[i,])-
    (velocityVectorY[j,]/velocityVectorX[j,]));
11.      end;
12. end;
```

**Kode Sumber 4.7 Implementasi Titik Potong**

#### 4.2.1.7 Implementasi Perhitungan Deteksi Situasi *False Conflict*

Pada implementasi *false conflict* juga akan diikut sertakan Implementasi jarak posisi pesawat(x,y) dengan atau tanpa sudut ke tititk potong. Implementasi disesuaikan dengan penjelasan teori *false conflict* pada sub-bab 2.2.2.5 dan perancangan deteksi *false conflict* pada sub-bab 3.3.2.7. Implementasi untuk deteksi situasi *false conflict* ditunjukkan pada Kode Sumber 4.8 dan Kode Sumber 4.9.

| | |
|---|---|
| Masukan | intersection point (intersectionPointInX, intersectionPointInY), sudut arah terbang(motion), koordinat x (koordinatX), dan koordinat y (koordinatY) |
| Keluaran | Vector kecepatan (velocityVectorX,velocityVectorY) dan intersection point (intersectionPointInX, intersectionPointInY) |

```
    /* distance intersection point*/
1.  do i = 1 to nrow(motion);
2.      do j=1  to nrow(motion);
3.              distanceInterToAircrafti[i,j] =
    sqrt(((koordinatX[i,]-
    intersectionPointInX[i,j])**2)+((koordinatY[i,]-
    intersectionPointInY[i,j])**2));
4.          distanceInterToAircraftj[i,j] = sqrt(((koordinatX[j,]-
    intersectionPointInX[i,j])**2)+((koordinatY[j,]-
    intersectionPointInY[i,j])**2));
5.          distanceInterToMotioni[i,j] =
    sqrt((((koordinatX[i,]+cosMotion[i,])-
    intersectionPointInX[i,j])**2)+(((koordinatY[i,]+sinMotion[i,])
    -intersectionPointInY[i,j])**2));
6.  distanceInterToMotionj[i,j] =
    sqrt((((koordinatX[j,]+cosMotion[j,])-
    intersectionPointInX[i,j])**2)+(((koordinatY[j,]+sinMotion[j,])
    -intersectionPointInY[i,j])**2));
7.      end;
8.  end;
    /* penyederhanaan rumus kriteria false conflict*/
9.  subtractDistancei= distanceInterToMotioni-
    distanceInterToAircrafti;
10. subtractDistancej= distanceInterToMotionj-
    distanceInterToAircraftj;do i = 1 to nrow(motion);
```

**Kode Sumber 4.8 Implementasi Deteksi Situasi *False Conflict* (Bagian 1)**

```
    /*False conflict detection*/
11. do i = 1 to nrow(motion);
12.     do j=1  to nrow(motion);
13.             if subtractDistancei[i,j]>0 &
    subtractDistancej[i,j]>0 & altitudeConflict[i,j] ^=1
    then falseConflict[i,j]= 1;
14.             else falseConflict[i,j] =0;
15.     end;
16. end;
```

**Kode Sumber 4.9 Implementasi Deteksi Situasi *False Conflict* (Bagian 2)**

#### 4.2.1.8 Implementasi Perhitungan Deteksi Kasus Anomali

Pada implementasi deteksi kasus anomali menggunakan koordinat x dan *safety distance*. Implementasi disesuaikan dengan penjelasan teori kasus anomali pada sub-bab 2.2.2.4 dan perancangan deteksi kasus anomali pada sub-bab 3.3.2.8. Implementasi untuk deteksi kasus anomali ditunjukkan pada Kode Sumber 4.10.

| Masukan | Koordinat X (koordinatX) dan *safety distance* (safetyDisatant) |
|---|---|
| Keluaran | Deteksi kasus anomali(pathologicalCase) |

```
1. do i = 1 to nrow(motion);
2.     do j=1  to nrow(motion);
3.         if abs(koordinatX[i,]-koordinatX[j,])<=
   safetyDistant[i,j] then pathologicalCase[i,j]=1;
4.             else pathologicalCase[i,j] =0;
5.     end;
6. end;
```

**Kode Sumber 4.10 Implementasi Deteksi Kasus Anomali**

#### 4.2.1.9 Implementasi Perhitungan Parameter Konstruksi Geometri

Pada implementasi parameter konstruksi geometri yang dimaksud adalah parameter $h_i, h_j, k_i, k_j$ dan $h'_i, h'_j, k'_i, k'_j$. Implementasi disesuaikan dengan penjelasan konstruksi geometri pada sub-

bab 2.2.2.3 untuk parameter $h_i, h_j, k_i, k_j$ dan sub-bab 2.2.2.4 untuk parameter $h'_i, h'_j, k'_i, k'_j$. Implementasi juga disesuaikan dengan perancangan sub-bab 3.3.2.9. Implementasi parameter konstruksi geometri ditunjukkan pada Kode Sumber 4.11.

| Masukan | sudut $l$ (sudutL), sudut $g$ (sudutG), sudut $\hat{l}$ (sudutLtop), sudut $\hat{g}$ (sudutGtop), dan *motion*/ sudut arah terbang (motion) |
|---|---|
| Keluaran | $h_i$ (Hi), $h_j$ (Hj), $k_i$ (Ki), $k_j$ (Kj), $h'_i$ (Hitop), $h'_j$ (Hjtop), $k'_i$ (Kitop), dan $k'_j$ (Kjtop) |

```
1.  do i = 1 to nrow(motion);
2.      do j=1  to nrow(motion);
3.              Hi[i,j] = (tan(sudutL[i,j])*cos(motion[i]))-
    (sin(motion[i])) ;
4.              Hj[i,j] = (tan(sudutL[i,j])*cos(motion[j]))-
    (sin(motion[j])) ;
5.            Ki[i,j] = (tan(sudutG[i,j])*cos(motion[i]))-
    (sin(motion[i])) ;
6.              Kj[i,j] = (tan(sudutG[i,j])*cos(motion[j]))-
    (sin(motion[j])) ;
7.              Hitop[i,j] = (-tan(sudutL[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
8.              Hjtop[i,j] = (-tan(sudutL[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
9.            Kitop[i,j] = (-tan(sudutG[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
10.             Kjtop[i,j] = (-tan(sudutG[i,j]+(pi/2))*
    sin(motion[i]))-(cos(motion[i])) ;
11.     end;
12. end;
```

**Kode Sumber 4.11 Implementasi Parameter Konstruksi Geometri**

### 4.2.2 Implementasi Model VAC untuk Penyelesaian Masalah Penghindaran Konflik

Berikut implementasi pencarian solusi penghindaran konflik menggunakan model VAC yang ditunjukkan oleh Kode Sumber 4.12, 4.13, 4.14, 4.15, dan 4.16. Implementasi model utama ini bertujuan untuk mendapatkan kecepatan optimal dengan menambahkan kecepatan awal dengan variasi kecepatan dan menempatkan pesawat di level ketinggian yang tidak akan

mengalami konflik dengan pesawat lainnya. Model utama menghasilkan variasi kecepatan *q*, level ketinggian setelah optimasi yang disimpan pada variabel *V_onthelevelZ* dan *altitudeLevel*, serta jumlah perubahan level yang dialami setiap pesawat *rho*. Implementasi sesuai dengan penjelasan pada Bab II dan juga disesuaikan dengan perancangan sub-bab 3.3.3.

| | |
|---|---|
| Masukan | $h_i$ (Hi), $h_j$ (Hj), $k_i$ (Ki), $k_j$ (Kj), $h'_i$ (Hitop), $h'_j$ (Hjtop), $k'_i$ (Kitop), $k'_j$ (Kjtop), Deteksi kasus anomali(pathologicalCase), deteksi false conflict (falseConflict), deteksi SC dan HTH (altitudeConflict), *motion/* sudut arah terbang (sudut), koordinat x (x), koordinat y (y), *velocity/*kecepatan (v), kecepatan maksimal (v_max), kecepatan minimal (v_min),level ketinggian pesawat (level), nilai biaya perubahan kecepatan positif (c_q_plus), nilai biaya perubahan kecepatan negatif (c_q_minus), nilai biaya perubahan ketinggian(c_j), banyaknya level ketinggian (z), nilai beban fungsi obyektif VC ($w_1$) , nilai beban fungsi obyektif VC ($w_2$) |
| Keluaran | Variasi kecepatan (q), variasi kecepatan positif (q_plus), variasi kecepatan negatif (q_minus), banyaknya perubahan level ketinggian (rho), level ketinggian setelah optimasi (V_onthelevelZ), variabel pemilihan model (delta_1, delta_2, delta_3, delta_4, delta 5) |

```
1.  set <num> Aircraft ;
2.  set <num> z = 1..4 ;
3.  number n;
4.  set<number> s = 1..n;
5.  number w{s};
6.  n = 2;
7.  w[1] = 0.5 ;
8.  w[2] =0.5 ;
9.  number levels{z};
10. number c_q_plus{Aircraft} = 1 ;
11. number c_q_minus{Aircraft} = 1 ;
12. number c_j{Aircraft}= 1;
```

**Kode Sumber 4.12 Implementasi Model VAC (Bagian 1)**

```
13. VAR q{Aircraft};
14. VAR q_plus{Aircraft}>=0 ;
15. VAR q_minus{Aircraft}>=0 ;
16. VAR alpha{Aircraft} BINARY;
17. VAR b{Aircraft} BINARY;
18. VAR rho{Aircraft} INTEGER >=0;
19. VAR beta{Aircraft}>=0;
20. VAR V_onthelevelZ{Aircraft,z} BINARY;
21. VAR delta_1 {i,j,z} BINARY;
22. VAR delta_2 {i,j,z} BINARY;
23. VAR delta_3 {i,j,z} BINARY;
24. VAR delta_4 {i,j,z} BINARY;
25. VAR delta_5 {i,j,z} BINARY;
26. VAR altitudeLevel{Aircraft} INTEGER >=0;


    *DECLARE OBJECTIVE*/
27. MIN VAC_model = w [1] * ( sum { f in Aircraft} ( (
    (c_q_plus[f] * q_plus[f])/ ( v_max [f] -  v_min [f] ) )
    + (c_q_minus[f] * q_minus[f])/ ( v_max [f] -  v_min [f]
    ) )) + w[2] * ( sum { f in Aircraft} ( c_j[f]*
    rho[f]));


    /* CONSTRAINT (10) */
28. CON CONSTRAINT_10 {f in Aircraft} :
    v_min[f] <= ( v[f] + q[f] ) <= v_max[f];/* CONSTRAINT
    (11) */


    /* CONSTRAINT (11) */
29. CON CONSTRAINT_11 { <f1,f2>  in {i,j}, le in z : f1 <
    f2 & falseConflict[f1,f2]=0} :
    ((v[f1] + q[f1]) * ((cos(sudut [f1]) * (1 -
    pathologicalCase[f1,f2])) - (sin(sudut[f1]) *
    pathologicalCase[f1,f2])))-(( v[f2] + q[f2] ) * ((
    cos(sudut[f2]) * ( 1 - pathologicalCase[f1,f2])) -
    (sin(sudut[f2]) * pathologicalCase[f1,f2] )))  <= ((
    v_max [f1] + v_max [f2] ) * (1 - delta_1 [f1,f2,le]));
```

**Kode Sumber 4.13 Implementasi Model VAC (Bagian 2)**

```
30. /* CONSTRAINT (12) */
    CON CONSTRAINT_12  { <f1,f2>  in {i,j}, le in z : f1 <
    f2 & falseConflict[f1,f2]=0}:
     - ((v[f1] + q[f1]) *((Hi[f1,f2]* ( 1 -
    pathologicalCase[f1,f2])) + (Hitop[f1,f2] *
    pathologicalCase[f1,f2]))  + ((v[f2] + q[f2])
    *((Hj[f1,f2]* ( 1 - pathologicalCase[f1,f2])) +
    (Hjtop[f1,f2] * pathologicalCase[f1,f2]))) <=
    (((((v_max [f1] * abs(Hi[f1,f2])) + ( v_max [f2] *
    abs(Hj[f1,f2]) )) * ( 1 - pathologicalCase[f1,f2])) +
    (((v_max [f1] * abs(Hitop [f1,f2])) + ( v_max [f2] *
    abs(Hjtop [f1,f2]))) * pathologicalCase[f1,f2])) * (1 -
    delta_1 [f1,f2,le]));

    /* CONSTRAINT (13) */
31. CON CONSTRAINT_13 { <f1,f2>  in {i,j}, le in z : f1 <
    f2  & falseConflict[f1,f2]=0}:
     ((v[f1] + q[f1]) * ((cos(sudut[f1]) * (1 -
    pathologicalCase[f1,f2])) - (sin(sudut[f1]) *
    pathologicalCase[f1,f2])))-(( v[f2] + q[f2] ) * ((
    cos(sudut [f2]) * ( 1 - pathologicalCase[f1,f2])) -
    (sin(sudut[f2]) * pathologicalCase[f1,f2] ))) <= ((
    v_max [f1] + v_max [f2] ) * (1 - delta_2 [f1,f2,le]));

    /* CONSTRAINT (14) */
32. CON CONSTRAINT_14 { <f1,f2>  in {i,j}, le in z: f1 < f2
    & falseConflict[f1,f2]=0}:
    ((v[f1] + q[f1]) *((Ki[f1,f2]* ( 1 -
    pathologicalCase[f1,f2])) + (Kitop[f1,f2] *
    pathologicalCase[f1,f2]))  - ((v[f2] + q[f2])
    *((Kj[f1,f2]* ( 1 - pathologicalCase[f1,f2])) +
    (Kjtop[f1,f2] * pathologicalCase[f1,f2]))) <=
    (((((v_max [f1] * abs(Ki[f1,f2])) + ( v_max [f2] *
    abs(Kj[f1,f2]) )) * ( 1 - pathologicalCase[f1,f2]))
    + (((v_max [f1] * abs(Kitop [f1,f2])) + ( v_max [f2] *
    abs(Kjtop [f1,f2]))) *  pathologicalCase[f1,f2])) * (1
    - delta_2 [f1,f2,le]));

    /* CONSTRAINT (15) */
33. CON CONSTRAINT_15 { <f1,f2>  in {i,j}, le in z: f1 < f2
    & falseConflict[f1,f2]=0}:
    -((v[f1] + q[f1]) * ((cos(sudut [f1]) * (1 -
    pathologicalCase[f1,f2])) - (sin(sudut [f1]) *
    pathologicalCase[f1,f2]))) +(( v[f2] + q[f2] ) * ((
    cos(sudut [f2]) * ( 1 - pathologicalCase[f1,f2])) -
    (sin(sudut[f2]) * pathologicalCase[f1,f2] ))) <= ((
    v_max [f1] + v_max [f2] ) * (1 - delta_3 [f1,f2,le]));
```

**Kode Sumber 4.14 Implementasi Model VAC (Bagian 3)**

```
    /* CONSTRAINT (16) */
34. CON CONSTRAINT_16 { <f1,f2>  in {i,j}, le in z: f1 < f2
    & falseConflict[f1,f2]=0}:
     ((v[f1] + q[f1]) *((Hi[f1,f2]* ( 1 -
    pathologicalCase[f1,f2])) + (Hitop[f1,f2] *
    pathologicalCase[f1,f2]))  - ((v[f2] + q[f2])
    *((Hj[f1,f2]* ( 1 - pathologicalCase[f1,f2])) +
    (Hjtop[f1,f2] * pathologicalCase[f1,f2]))) <=
    (((((v_max [f1] * abs(Hi[f1,f2])) + ( v_max [f2] *
    abs(Hj[f1,f2]) )) * ( 1 - pathologicalCase[f1,f2])) +
    (((v_max [f1] * abs(Hitop [f1,f2])) + ( v_max [f2] *
    abs(Hjtop [f1,f2]))) *  pathologicalCase[f1,f2])) * (1
    - delta_3 [f1,f2,le]));

    /* CONSTRAINT (17) */
35. CON CONSTRAINT_17  { <f1,f2>  in {i,j}, le in z : f1 <
    f2  & falseConflict[f1,f2]=0}:
    -((v[f1] + q[f1]) * ((cos(sudut [f1]) * (1 -
    pathologicalCase[f1,f2])) - (sin(sudut [f1]) *
    pathologicalCase[f1,f2]))) +(( v[f2] + q[f2] ) * ((
    cos(sudut [f2]) * ( 1 - pathologicalCase[f1,f2])) -
    (sin(sudut[f2]) * pathologicalCase[f1,f2] ))) <= ((
    v_max [f1] + v_max [f2] ) * (1 - delta_4 [f1,f2,le]));

    /* CONSTRAINT (18) */
36. CON CONSTRAINT_18  { <f1,f2>  in {i,j}, le in z: f1 <
    f2 & falseConflict[f1,f2]=0}:
    - ((v[f1] + q[f1]) *((Ki[f1,f2]* ( 1 -
    pathologicalCase[f1,f2])) + (Kitop[f1,f2] *
    pathologicalCase[f1,f2]))  + ((v[f2] + q[f2])
    *((Kj[f1,f2]* ( 1 - pathologicalCase[f1,f2])) +
    (Kjtop[f1,f2] * pathologicalCase[f1,f2]))) <=
    (((((v_max [f1] * abs(Ki[f1,f2])) + ( v_max [f2] *
    abs(Kj[f1,f2]) )) * ( 1 - pathologicalCase[f1,f2])) +
    (((v_max [f1] * abs(Kitop [f1,f2])) + ( v_max [f2] *
    abs(Kjtop [f1,f2]))) *  pathologicalCase[f1,f2])) * (1
    - delta_4 [f1,f2,le]));

    /* CONSTRAINT (19) */
37. CON CONSTRAINT_19 { <f1,f2>  in {i,j}, le in z: f1 < f2
    & falseConflict[f1,f2]=0 }:
    delta_1 [f1,f2,le] + delta_2 [f1,f2,le] + delta_3
    [f1,f2,le] + delta_4 [f1,f2,le] + delta_5 [f1,f2,le] =
    1;
```

**Kode Sumber 4.15  Implementasi Model VAC (Bagian 4)**

```
    /* CONSTRAINT (20) */
38. CON CONSTRAINT_20 { <f1,f2>  in {i,j}, le in z : f1 <
    f2 }:
    altitudeConflict[f1,f2] <= altitudeConflict[f1,f2] *
    delta_5 [f1,f2,le] ;

    /* CONSTRAINT (21) */
39. CON CONSTRAINT_21 { <f1,f2>  in {i,j}, le in z : f1 <
    f2}:
    V_onthelevelZ [f1,le] + V_onthelevelZ [f2,le] >=
    delta_5 [f1,f2,le] - 1 ;

    /* CONSTRAINT (22) */
40. CON CONSTRAINT_22 { <f1,f2>  in {i,j}, le in z : f1 <
    f2}:
    V_onthelevelZ [f1,le] + V_onthelevelZ [f2,le] <= 2 -
    delta_5 [f1,f2,le] ;

    /* CONSTRAINT (23) */
41. CON CONSTRAINT_23 {f in Aircraft}:
    sum { le in z} V_onthelevelZ [f,le] = 1 ;

    /* CONSTRAINT (27) */
42. CON CONSTRAINT_27 {f in Aircraft}:
    (sum { le in z} ( le* V_onthelevelZ[f, le])) - level[f]
    <= rho[f];

    /* CONSTRAINT (28) */
43. CON CONSTRAINT_28 {f in Aircraft} :
    level[f] - (sum { le in z} ( le* V_onthelevelZ[f,le]))
    <= rho[f];

44. CON CONSTRAINT_33_2 {f in Aircraft } :
    q[f] = q_plus[f] - q_minus[f];

45. CON CONSTRAINT_33{f in Aircraft } :
    (sum { le in z} ( le* V_onthelevelZ[f, le]))=
    altitudeLevel[f];

46. SOLVE;
47. print q q_plus q_minus rho  altitudeLevel;
```

**Kode Sumber 4.16 Implementasi Model VAC (Bagian 5)**

# BAB V
# HASIL UJI COBA DAN EVALUASI

Bab ini merupakan bahasan mengenai hasil uji coba dan evaluasi terhadap model VAC untuk menyelesaikan permasalahan penghidaran konflik. Pembahasan pada bab ini meliputi lingkungan uji coba, skenario uji coba, hasil uji coba, dan evaluasi.

## 5.1 Lingkungan Pelaksanaan Uji Coba

Spesifikasi perangkat keras dan lunak yang digunakan dalam uji coba aplikasi dapat dilihat pada Tabel 5.1.

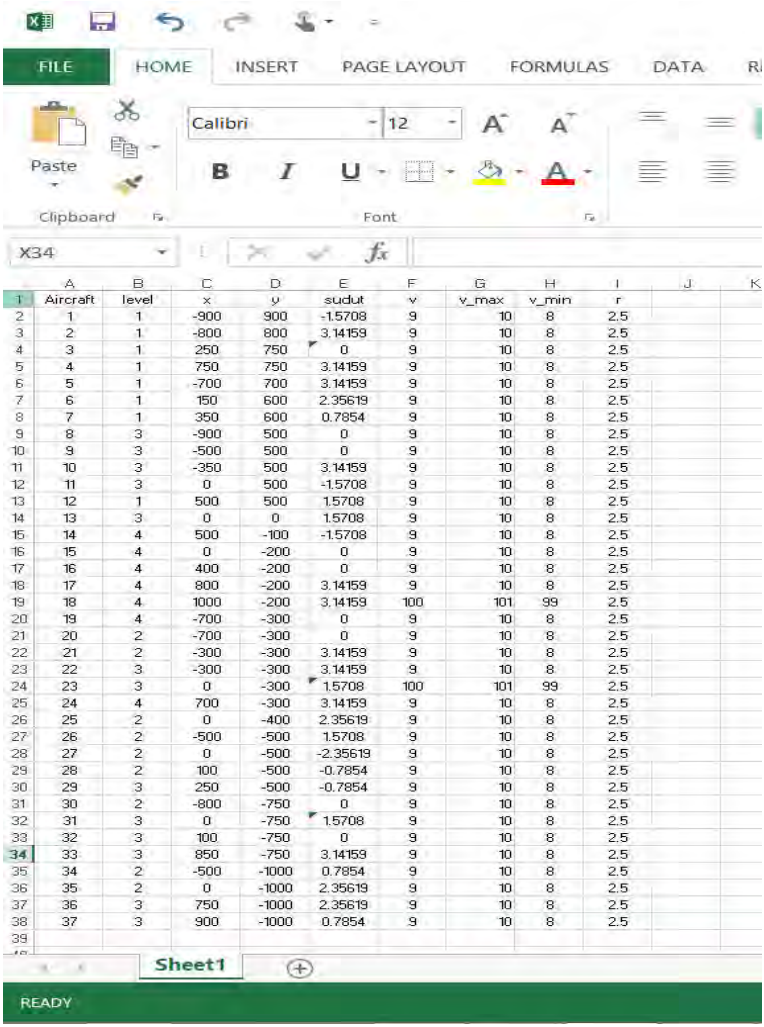
**Tabel 5.1 Lingkungan Uji Coba**

| | | |
|---|---|---|
| Perangkat Keras | Prosesor | Intel Core i7-2350M CPU @ 2.30 Ghz |
| | Memori RAM | 4 GB |
| Perangkat Lunak | Sistem Operasi | Microsoft Windows 8 Pro 64-*bit* |
| | Perangkat Pengembang | SAS 9.2 |
| | | Microsoft Excel |

## 5.2 Data Uji Coba

Data masukan direpresentasikan dalam file dengan format teks. Representasi data masukan pada Microsoft Excel dapat dilihat pada Gambar 5.1. Data uji coba yang digunakan pada tahapan uji coba adalah data uji coba yang digunakan oleh Alonso et al [1] (selanjutnya disebut data uji coba utama). Terdapat dua data uji coba yang digunakan uji kinerja dan uji kebenaran. Pada uji kinerja, data uji coba yang digunakan adalah data uji coba 37 pesawat dan untuk uji kebenaran menggunakan dua data yaitu data uji coba 10 dengan model VC tanpa kondisi *false conflict* dan kasus anomali serta data uji coba 37 pesawat dengan model VAC . Data 37 pesawat dapat

dilihat pada Tabel 3.1 di Bab III dan data 10 pesawat dapat dilihat pada Tabel 5.2.



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Aircraft | level | x | y | sudut | v | v_max | v_min | r |
| 2 | 1 | 1 | -900 | 900 | -1.5708 | 9 | 10 | 8 | 2.5 |
| 3 | 2 | 1 | -800 | 800 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 4 | 3 | 1 | 250 | 750 | 0 | 9 | 10 | 8 | 2.5 |
| 5 | 4 | 1 | 750 | 750 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 6 | 5 | 1 | -700 | 700 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 7 | 6 | 1 | 150 | 600 | 2.35619 | 9 | 10 | 8 | 2.5 |
| 8 | 7 | 1 | 350 | 600 | 0.7854 | 9 | 10 | 8 | 2.5 |
| 9 | 8 | 3 | -900 | 500 | 0 | 9 | 10 | 8 | 2.5 |
| 10 | 9 | 3 | -500 | 500 | 0 | 9 | 10 | 8 | 2.5 |
| 11 | 10 | 3 | -350 | 500 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 12 | 11 | 3 | 0 | 500 | -1.5708 | 9 | 10 | 8 | 2.5 |
| 13 | 12 | 1 | 500 | 500 | 1.5708 | 9 | 10 | 8 | 2.5 |
| 14 | 13 | 3 | 0 | 0 | 1.5708 | 9 | 10 | 8 | 2.5 |
| 15 | 14 | 4 | 500 | -100 | -1.5708 | 9 | 10 | 8 | 2.5 |
| 16 | 15 | 4 | 0 | -200 | 0 | 9 | 10 | 8 | 2.5 |
| 17 | 16 | 4 | 400 | -200 | 0 | 9 | 10 | 8 | 2.5 |
| 18 | 17 | 4 | 800 | -200 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 19 | 18 | 4 | 1000 | -200 | 3.14159 | 100 | 101 | 99 | 2.5 |
| 20 | 19 | 4 | -700 | -300 | 0 | 9 | 10 | 8 | 2.5 |
| 21 | 20 | 2 | -700 | -300 | 0 | 9 | 10 | 8 | 2.5 |
| 22 | 21 | 2 | -300 | -300 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 23 | 22 | 3 | -300 | -300 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 24 | 23 | 3 | 0 | -300 | 1.5708 | 100 | 101 | 99 | 2.5 |
| 25 | 24 | 4 | 700 | -300 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 26 | 25 | 2 | 0 | -400 | 2.35619 | 9 | 10 | 8 | 2.5 |
| 27 | 26 | 2 | -500 | -500 | 1.5708 | 9 | 10 | 8 | 2.5 |
| 28 | 27 | 2 | 0 | -500 | -2.35619 | 9 | 10 | 8 | 2.5 |
| 29 | 28 | 2 | 100 | -500 | -0.7854 | 9 | 10 | 8 | 2.5 |
| 30 | 29 | 3 | 250 | -500 | -0.7854 | 9 | 10 | 8 | 2.5 |
| 31 | 30 | 2 | -800 | -750 | 0 | 9 | 10 | 8 | 2.5 |
| 32 | 31 | 3 | 0 | -750 | 1.5708 | 9 | 10 | 8 | 2.5 |
| 33 | 32 | 3 | 100 | -750 | 0 | 9 | 10 | 8 | 2.5 |
| 34 | 33 | 3 | 850 | -750 | 3.14159 | 9 | 10 | 8 | 2.5 |
| 35 | 34 | 2 | -500 | -1000 | 0.7854 | 9 | 10 | 8 | 2.5 |
| 36 | 35 | 2 | 0 | -1000 | 2.35619 | 9 | 10 | 8 | 2.5 |
| 37 | 36 | 3 | 750 | -1000 | 2.35619 | 9 | 10 | 8 | 2.5 |
| 38 | 37 | 3 | 900 | -1000 | 0.7854 | 9 | 10 | 8 | 2.5 |
| 39 | | | | | | | | | |

**Gambar 5.1 Representasi Data Masukan Dalam Microsoft Excel**

**Tabel 5.2 Data 10 Pesawat**

| Pesawat | Koordinat X | Koordinat Y | Sudut Arah Terbang | Kecepatan | Kecepatan Maksimal | Kecepatan Minimal | Safety radius |
|---|---|---|---|---|---|---|---|
| 1 | -300 | 550 | -1.571 | 4 | 5 | 3 | 2.5 |
| 2 | 350 | 300 | 0 | 4 | 5 | 3 | 2.5 |
| 3 | -500 | 150 | 0.7854 | 4 | 5 | 3 | 2.5 |
| 4 | -100 | 150 | 2.3562 | 4 | 5 | 3 | 2.5 |
| 5 | 550 | 100 | 1.5708 | 4 | 5 | 3 | 2.5 |
| 6 | 150 | -200 | -0.785 | 4 | 5 | 3 | 2.5 |
| 7 | 55 | -20 | -2.356 | 4 | 5 | 3 | 2.5 |
| 8 | -30 | -35 | 3.1416 | 4 | 5 | 3 | 2.5 |
| 9 | -50 | -55 | 1.5708 | 4 | 5 | 3 | 2.5 |
| 10 | 35 | -60 | 1.5708 | 4 | 5 | 3 | 2.5 |

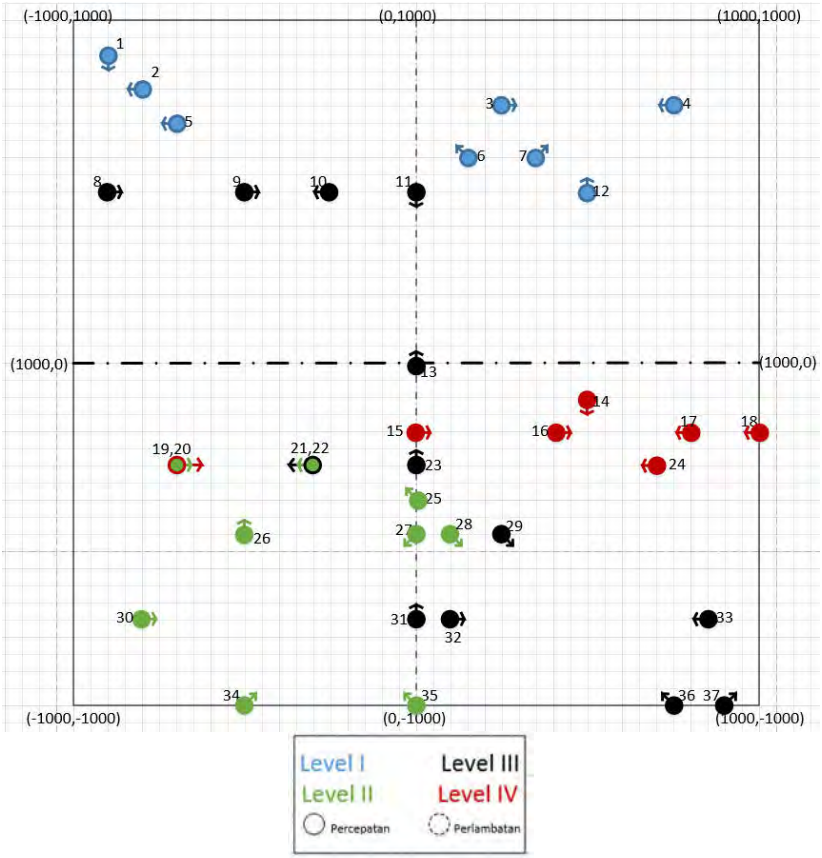
## 5.3 Skenario Uji Coba

Pada bagian ini dijelaskan mengenai rangkaian skenario uji coba yang telah dilakukan. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja aplikasi.

Hasil uji coba akan menampilkan tiga hasil yaitu kesimpulan masalah pada rancangan model (*problem summary*), informasi performa computer (*performance information*), kesimpulan hasil optimasi (*solution summary*), dan hasil optimasi (*optimization result*) yang menampilkan nilai variabel keputusan.

### 5.3.1 Uji Kinerja

Tahap ini akan menguji kinerja program terhadap waktu kerja program untuk mendapatkan solusi. Program model VAC akan di jalankan sebanyak sepuluh kali dan hasil kinerja waktu optimasi adalah rata rata dari sepuluh waktu uji coba. Waktu uji coba terbagi dua yaitu waktu cpu (*cpu time*) dan waktu menampikan hasil (*real time*).

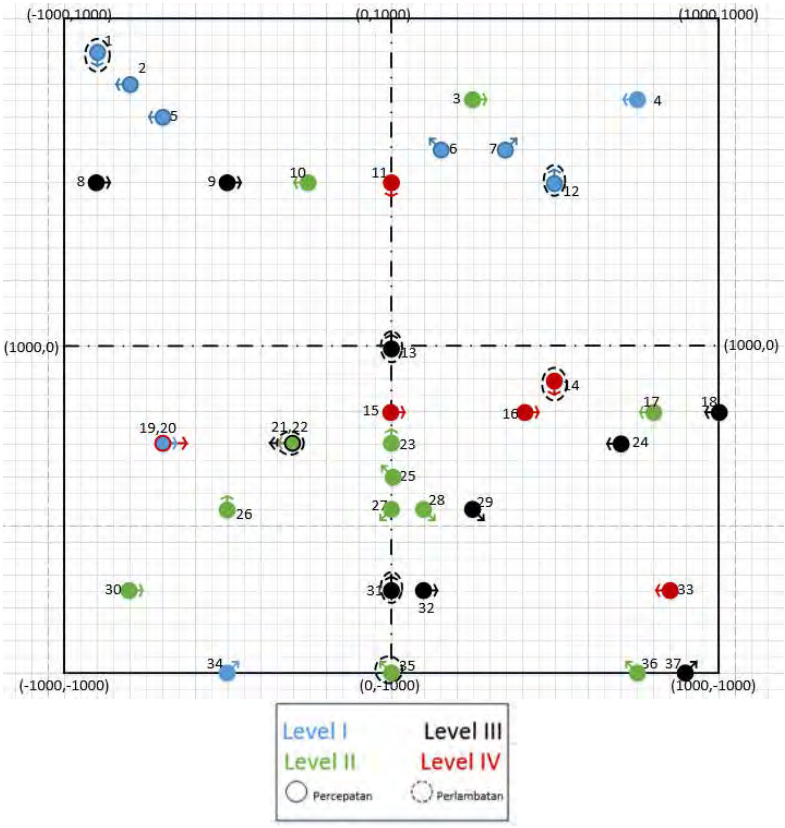



**Gambar 5.2 Visualisasi Data Uji Coba 37 Pesawat**

### 5.3.1.1 Uji Kinerja 37 Pesawat

Sub bab ini menunjukkan uji kinerja program terhadap waktu kerja pengujian 37 pesawat. Data uji coba 37 pesawat dapat dilihat pada Tabel 3.1 di Bab III dan visualisasi data uji dapat dilihat pada Gambar 5.2. Hasil optimasi untuk data uji coba 37 pesawat dapat dilihat pada Gambar 3.1 dan 3.2. Gambar 3.1 dan 3.2 menampilkan *optimization result*. Pada optimization result yang akan

ditampilkan adalah nilai variabel *q, q_plus, q_minus, altitudeLevel*, dan *VonthelevelZ*. Untuk waktu proses optimasi, perbandingan hasil penulis dengan Alonso et al [1], *problem summary*, *performance information,* dan *solution summary* akan ditampilkan pada Lampiran A. Untuk melihat kinerja waktu, uji coba dilakukan sebanyak 10 kali terhadap data pesawat dengan label *TESTESIS2.xls*. Visualisasi hasil uji coba oleh penulis dapat dilihat pada Gambar 5.3.




**Gambar 5.3 Visualisasi Hasil Optimasi pada Uji Coba 37 Pesawat**

Hasil uji coba 37 pesawat menghasilkan nilai obyektif 6.5562054543 atau dibulatkan menjadi 6.5562 dengan jumlah *constraint* sebanyak 29130, 27602 variabel *binary* dan 37 variabel *continue*. Untuk hasil kinerjanya menghasilkan waktu *cpu time* 21 menit 19,15 detik dan *real time* 21 menit 46,08 detik

### 5.3.2 Uji Kebenaran

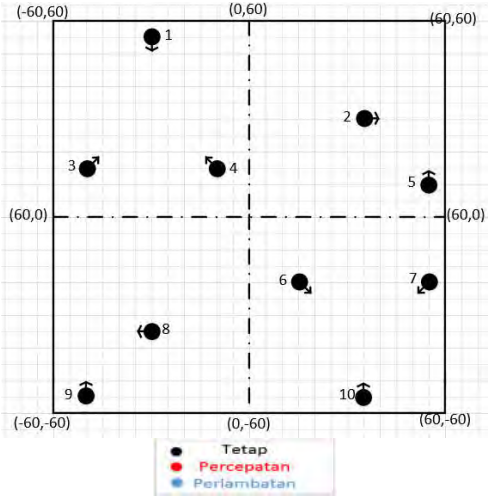
Pada bagian ini akan dilakukan perbandingan hasil program optimasi yang dikembangkan dengan kriteria pembanding yang menjadi tujuan dari penyelesaian permasalahan penghindaran konflik.

Uji kebenaran ini akan menguji kesesuaian model VAC yang dikembangkan oleh penulis dengan model VAC yang ada dikembangkan oleh Alonso et al [1]. Uji kebenaran ini dilakukan pada 10 pesawat dengan menggunakan model VC tanpa *false conflict* dan kasus anomali.

#### 5.3.2.1 Uji Kebenaran 10 Pesawat

Sub-bab ini menunjukkan uji kebeneran 10 pesawat dengan menggunakan model VC tanpa *false conflict* dan kasus anomali. Data uji coba 10 pesawat dapat dilihat pada Tabel 5.2 dengan *label TA-UjiVELO.xls*. Visualisasi data uji dapat dilihat pada Gambar 5.4. Hasil uji coba dengan model VC oleh penulis ditunjukkan pada Gambar 5.5 dan Gambar 5.6 serta pada Tabel 5.8. Pada Gambar 5.5 dan Gambar 5.6 menampilkan variabel keputusan delta, pada gambar ditunjukkan pasangan pesawat yaitu pesawat *i* dan pesawat *j* dan delta sebagai hasil kepusan dari batasan konstruksi geometri (2.8). Tabel 5.8 menampilkan *optimization result* variasi kecepatan. Dari hasil uji coba yang dilakukan penulis dibandingkan dengan hasil uji coba yang dilakukan oleh Alonso et al [1]. Perbandingan dapat dilihat pada Gambar 5.7 untuk visualisasi hasil uji coba Alonso et al [1] dan Gambar 5.8 untuk hasil uji coba oleh penulis. Untuk *problem summary*, *performance information,* dan *solution summary* akan ditampilkan pada Lampiran A.

Hasil uji coba 10 pesawat menghasilkan nilai obyektif 5.298889578 atau dibulatkan menjadi 5.299 dengan jumlah *constraint* sebanyak 425, 400 variabel *binary*, 10 variabel *continue, cpu time* 0.60 detik, dan *real time* 0.87 detik.




**Gambar 5.4 Visualisasi Data Uji Coba 10 Pesawat**

**Tabel 5.3 *Optimization Result* Hasil Variasi Kecepatan Uji Coba 10 Pesawat**

| Pesawat | Variasi Kecepatan |
|---|---|
| **1** | 0.881524753 |
| **2** | -1 |
| **3** | 0 |
| **4** | -0.89512954 |
| **5** | 0.313552873 |
| **6** | -0.89512954 |
| **7** | 0 |
| **8** | 0.313552873 |
| **9** | -1 |
| **10** | 0 |

| Pesawat *i* | Pesawat *j* | delta_1 | delta_2 | delta_3 | delta_4 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 0 | 0 | 0 |
| 1 | 3 | 1 | 0 | 0 | 0 |
| 1 | 4 | 0 | 0 | 0 | 1 |
| 1 | 5 | 1 | 0 | 0 | 0 |
| 1 | 6 | 0 | 0 | 1 | 0 |
| 1 | 7 | 1 | 0 | 0 | 0 |
| 1 | 8 | 0 | 0 | 1 | 0 |
| 1 | 9 | 1 | 0 | 0 | 0 |
| 1 | 10 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 |
| 2 | 3 | 0 | 0 | 0 | 1 |
| 2 | 4 | 0 | 0 | 0 | 1 |
| 2 | 5 | 0 | 0 | 0 | 1 |
| 2 | 6 | 0 | 0 | 0 | 1 |
| 2 | 7 | 0 | 0 | 1 | 0 |
| 2 | 8 | 0 | 0 | 0 | 1 |
| 2 | 9 | 0 | 0 | 0 | 1 |
| 2 | 10 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 3 | 2 | 0 | 0 | 0 | 0 |
| 3 | 3 | 0 | 0 | 0 | 0 |
| 3 | 4 | 0 | 0 | 1 | 0 |
| 3 | 5 | 0 | 0 | 0 | 1 |
| 3 | 6 | 0 | 0 | 1 | 0 |
| 3 | 7 | 0 | 1 | 0 | 0 |
| 3 | 8 | 0 | 0 | 1 | 0 |
| 3 | 9 | 0 | 0 | 1 | 0 |
| 3 | 10 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 |
| 4 | 2 | 0 | 0 | 0 | 0 |
| 4 | 3 | 0 | 0 | 0 | 0 |
| 4 | 4 | 0 | 0 | 0 | 0 |
| 4 | 5 | 1 | 0 | 0 | 0 |
| 4 | 6 | 0 | 1 | 0 | 0 |
| 4 | 7 | 0 | 1 | 0 | 0 |
| 4 | 8 | 0 | 0 | 0 | 1 |
| 4 | 9 | 0 | 1 | 0 | 0 |
| 4 | 10 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 5 | 2 | 0 | 0 | 0 | 0 |
| 5 | 3 | 0 | 0 | 0 | 0 |
| 5 | 4 | 0 | 0 | 0 | 0 |
| 5 | 5 | 0 | 0 | 0 | 0 |
| 5 | 6 | 0 | 0 | 1 | 0 |
| 5 | 7 | 1 | 0 | 0 | 0 |
| 5 | 8 | 0 | 0 | 1 | 0 |
| 5 | 9 | 0 | 0 | 1 | 0 |
| 5 | 10 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 |
| 6 | 2 | 0 | 0 | 0 | 0 |
| 6 | 3 | 0 | 0 | 0 | 0 |
| 6 | 4 | 0 | 0 | 0 | 0 |
| 6 | 5 | 0 | 0 | 0 | 0 |
| 6 | 6 | 0 | 0 | 0 | 0 |
| 6 | 7 | 0 | 1 | 0 | 0 |
| 6 | 8 | 0 | 0 | 0 | 1 |
| 6 | 9 | 1 | 0 | 0 | 0 |
| 6 | 10 | 1 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 7 | 2 | 0 | 0 | 0 | 0 |
| 7 | 3 | 0 | 0 | 0 | 0 |
| 7 | 4 | 0 | 0 | 0 | 0 |
| 7 | 5 | 0 | 0 | 0 | 0 |
| 7 | 6 | 0 | 0 | 0 | 0 |
| 7 | 7 | 0 | 0 | 0 | 0 |
| 7 | 8 | 0 | 0 | 0 | 1 |
| 7 | 9 | 0 | 0 | 0 | 1 |
| 7 | 10 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 8 | 2 | 0 | 0 | 0 | 0 |
| 8 | 3 | 0 | 0 | 0 | 0 |
| 8 | 4 | 0 | 0 | 0 | 0 |
| 8 | 5 | 0 | 0 | 0 | 0 |
| 8 | 6 | 0 | 0 | 0 | 0 |
| 8 | 7 | 0 | 0 | 0 | 0 |
| 8 | 8 | 0 | 0 | 0 | 0 |
| 8 | 9 | 0 | 1 | 0 | 0 |
| 8 | 10 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 9 | 2 | 0 | 0 | 0 | 0 |
| 9 | 3 | 0 | 0 | 0 | 0 |
| 9 | 4 | 0 | 0 | 0 | 0 |
| 9 | 5 | 0 | 0 | 0 | 0 |
| 9 | 6 | 0 | 0 | 0 | 0 |
| 9 | 7 | 0 | 0 | 0 | 0 |
| 9 | 8 | 0 | 0 | 0 | 0 |
| 9 | 9 | 0 | 0 | 0 | 0 |
| 9 | 10 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 | 0 |
| 10 | 2 | 0 | 0 | 0 | 0 |

**Gambar 5.5 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 1)**

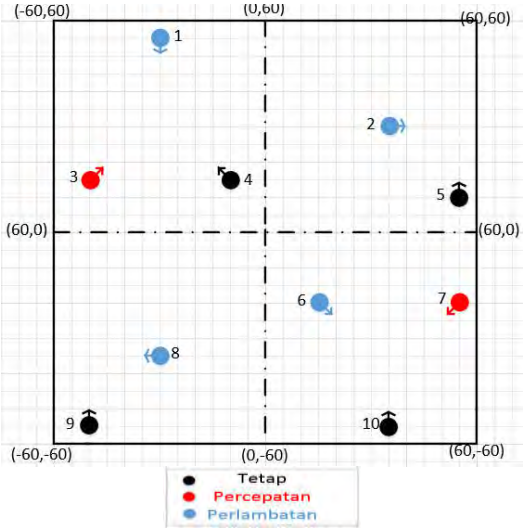
| Pesawat *i* | Pesawat *j* | delta_1 | delta_2 | delta_3 | delta_4 |
|---|---|---|---|---|---|
| 10 | 3 | 0 | 0 | 0 | 0 |
| 10 | 4 | 0 | 0 | 0 | 0 |
| 10 | 5 | 0 | 0 | 0 | 0 |
| 10 | 6 | 0 | 0 | 0 | 0 |
| 10 | 7 | 0 | 0 | 0 | 0 |
| 10 | 8 | 0 | 0 | 0 | 0 |
| 10 | 9 | 0 | 0 | 0 | 0 |
| 10 | 10 | 0 | 0 | 0 | 0 |

**Gambar 5.6 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 2)**

Perbedaan hasil uji coba penulis dengan hasil uji coba oleh Alonso et al [1] adalah

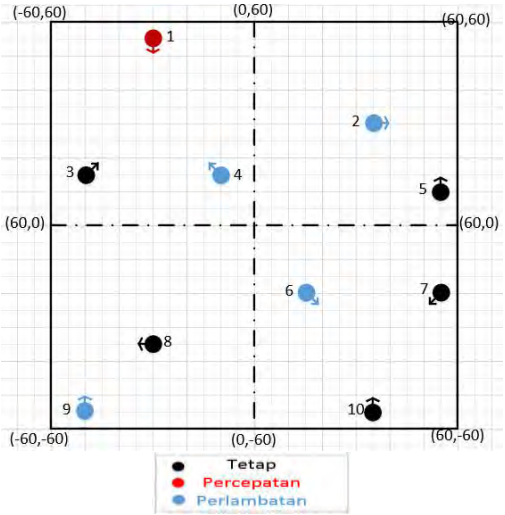
1. Pesawat 1,3, dan 4 membentuk segitiga sama kaki, sehingga konflik terjadi antara pesawat 3 dan pesawat 4. Pesawat 1 tidak mengalami konflik dengan pesawat 3 maupun dengan pesawat 4. Pada hasil uji coba Alonso et al [1], pesawat 1 mengalami perlambatan, pesawat 3 percepatan dan pesawat 4 tidak mengalami perubahan. Pada hasil uji coba penulis, pesawat 1 mengalami percepatan, pesawat 3 tidak mengalami perubahan dan pesawat 4 tidak mengalami perlambatan. Sehingga perbandingan hasil uji coba antara penulis dan Alonso et al [1] untuk pesawat 1,3, dan 4 adalah sama karena hasil uji coba hanya tertukar satu sama lain.
2. Pesawat 9 dan 8 saling mengalami konflik satu sama lain. Pada hasil uji coba Alonso et al [1], pesawat 9 mengalami perlambatan dan pesawat 8 tidak mengalami perubahan kecepatan. Pada hasil uji coba penulis, pesawat 9 tidak mengalami perubahan kecepatan dan pesawat 8 mengalami perlambatan. Sehingga perbandingan hasil uji coba antara penulis dan Alonso et al [1] untuk pesawat 9 dan 8 adalah sama karena hasil uji coba hanya tertukar satu sama lain.
3. Pesawat 6, 7, dan 10 membentuk segitiga sama kaki, sehingga konflik terjadi antara pesawat 6 dan pesawat 7. Pesawat 10 tidak mengalami konflik dengan pesawat 6 maupun dengan pesawat 7. Pada hasil uji coba Alonso et al

[1], pesawat 6 mengalami perlambatan, pesawat 7 percepatan dan pesawat 10 tidak mengalami perubahan. Pada hasil uji coba penulis, pesawat 6 mengalami perlambatan, pesawat 7 dan pesawat 10 tidak mengalami perlambatan, sehingga pesawat 6 meberikan jalan terlebih dahulu untuk pesawat 7. Konflik tidak terjadi pada pesawat 10 karena perlambatan kecepatan pesawat 6 tidak mempengaruhi jalannya pesawat 10, sehingga hasil uji coba yang dilakukan penulis pada pesawat 6, 7, dan 10 tidak mengalami konflik.




**Gambar 5.7 Visualisasi Hasil Uji Coba 10 Pesawat oleh Alonso**

Kesimpulannya adalah hasil uji coba yang dilakukan oleh penulis sama benarnya dengan hasil uji coba yang dilakukan oleh Alonso et al [1] dikarenakan semua konflik terjadi pada sebelum optimasi telah dihindari dengan baik.




**Gambar 5.8 Visualisasi Hasil Uji Coba 10 Pesawat oleh Penulis dengan Threshold 0.35**

*[Halaman ini sengaja dikosongkan]*

# LAMPIRAN A
# HASIL UJI

Lampiran A berisikan screenshoot hasil *cpu time*, *real time* dan perbandingan hasil.

## 1. *Screenshot* Hasil *Cpu Time* dan *Real Time*

Pada bagian ini akan berisikan *screenshot* hasil *cpu time* dan *real time* uji kinerja yang sudah dijelaskan pada sub bab 5.3.1

```
811  print q q_plus q_minus rho altitudeLevel V_onthelevelZ delta_1 delta_2 delta_3 delta_4
811! delta_5;
812  quit;
NOTE: PROCEDURE OPTMODEL used (Total process time):
      real time           21:46.08
      cpu time            21:19.15
```

**Gambar A.1 Hasil Uji Kinerja**

## *2.* Hasil Perbandingan Uji Coba 37 Pesawat

Pada bagian ini akan menjelaskan perbandingan hasil uji coba penulis dengan hasil uji coba yang dilakukan oleh Alonso Alonso et al [1]. Uji coba ini menggunakan model VAC. Visualisasi hasil uji coba yang dilakukan oleh Alonso et al [1] dapat dilihat pada Gambar A.11, hasil uji coba penulis dapat dilihat pada Gambar 5.5.3, dan dataset awal dapat dilihat pada Gambar 5.5.2 .

Hasil uji coba yang dilakukan oleh Alonso et al [1] dan penulis mempunyai nilai obyektif yang sama yaitu 6.5562. Perbandingan hasil penulis dan Alonso et al [1] adalah sebagai berikut

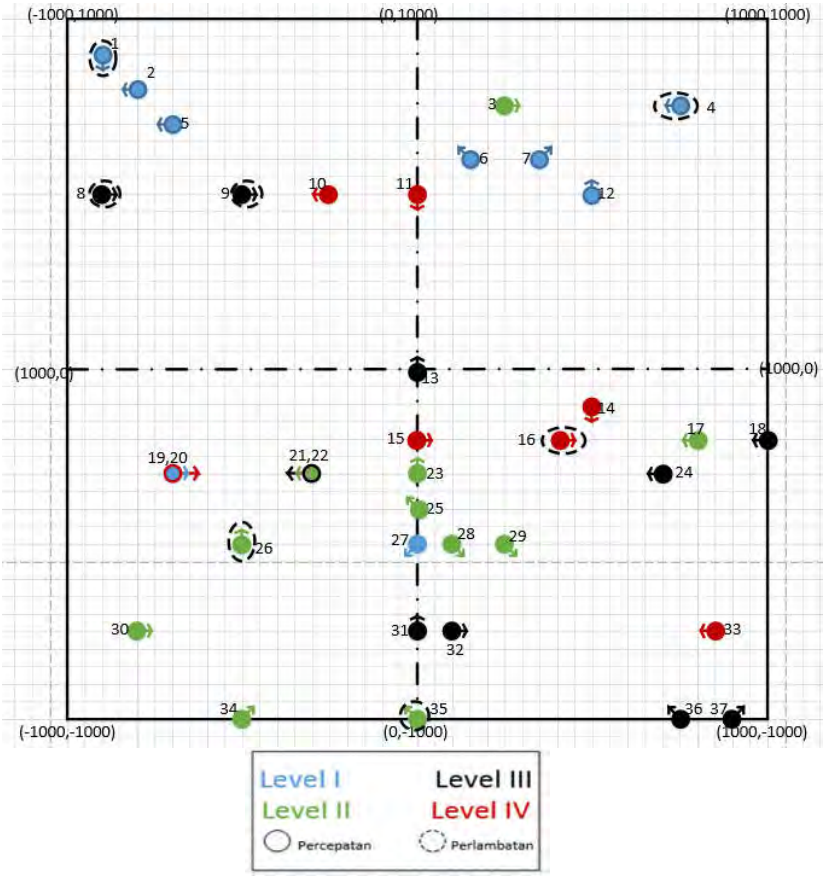
1. Pesawat 9 dan 13 mengalami konflik. Pada hasil uji coba Alonso et al [1] solusi atas permasalahan tersebut dengan memperlambat kecepatan pesawat 9 dan pesawat 8. Hal ini dikarenakan pesawat 9 memperlambat kecepatan, pesawat dibelakang pesawat 9 juga harus memperlambat kecepatan

yaitu pesawat 8. Pada hasil uji coba penulis, pesawat 13 dan 31 yang memperlambat kecepatan. Hasil penulis untuk konflik pesawat 9 dan pesawat 13 sama baiknya dengan hasil Alonso et al [1] dikarenakan hanya tertukar untuk pesawat yang mengalami perlambatan kecepatan.

2. Pesawat 14 dan 16 mengalami konflik. Pada hasil uji coba Alonso et al [1] solusi atas permasalahan tersebut dengan memperlambat kecepatan pesawat 16. Pada kasus ini pesawat 15 tidak mengalami perlambatan, seharusnya jika pesawat 16 mengalami perlambatan pesawat dibelakangnya dan dalam satu level yang sama harus memperlambat kecepatannya juga. Pada hasil uji coba penulis, pesawat 14 yang memperlambat kecepatan. Hasil penulis untuk konflik pesawat 14 dan pesawat 16 lebih baik dibandingkan hasil Alonso et al [1] dikarenakan hasil Alonso tidak menyelesaikan konflik yang terjadi.
3. Pesawat 4 dan 6 mengalami konflik. Pada hasil Alonso et al [1] pesawat 4 yang mengalami perlambatan. Pada hasil penulis pesawat 6 yang mengalami perlambatan. Hasil penulis untuk konflik pesawat 4 dan pesawat 6 sama baiknya dengan hasil Alonso et al [1] dikarenakan hanya tertukar untuk pesawat yang mengalami perlambatan kecepatan.
4. Pesawat 21 dan 26 mengalami konflik. Pada hasil Alonso et al [1] pesawat 26 yang mengalami perlambatan. Pada hasil penulis pesawat 21 yang mengalami perlambatan. Hasil penulis untuk konflik pesawat 21 dan pesawat 26 sama baiknya dengan hasil Alonso et al [1] dikarenakan hanya tertukar untuk pesawat yang mengalami perlambatan kecepatan.
5. Pesawat 27 dan 34 mengalami konflik *head to head*. Pada hasil Alonso et al [1] pesawat 27 yang mengalami perubahan ketinggian yaitu turun satu level menjadi di level 1. Pada hasil penulis pesawat 34 yang mengalami perubahan ketinggian yaitu turun satu level menjadi di level 1. Hasil

penulis untuk konflik pesawat 27 dan pesawat 34 sama baiknya dengan hasil Alonso et al [1] dikarenakan hanya tertukar untuk pesawat yang mengalami perubahan level ketinggian dan perubahan sama-sama turun satu level.

6. Pesawat 29 dan 36 mengalami konflik *head to head*. Pada hasil Alonso et al [1] pesawat 29 yang mengalami perubahan ketinggian yaitu naik satu level menjadi di level 3. Pada hasil penulis pesawat 36 yang mengalami perubahan ketinggian yaitu naik satu level menjadi di level 3. Hasil penulis untuk konflik pesawat 27 dan pesawat 34 sama baiknya dengan hasil Alonso et al [1] dikarenakan hanya tertukar untuk pesawat yang mengalami perubahan level ketinggian dan perubahan sama-sama naik satu level.
7. Pesawat 8,9 dan 10 mengalami konflik *head to head*. Pada hasil Alonso et al [1] pesawat 10 mengalami perubahan ketinggian yaitu naik satu level menjadi di level 4. Pada hasil penulis pesawat 10 mengalami perubahan ketinggian yaitu turun satu level menjadi di level 2. Hasil penulis untuk konflik pesawat 27 dan pesawat 34 sama baiknya dengan hasil Alonso et al [1] dikarenakan hanya tertukar untuk pesawat yang mengalami perubahan level ketinggian dan terhindar dari konflik *head to head*.




**Gambar A.2 Visualisasi Hasil Uji Coba oleh Alonso**

### *3. Hasil Uji SAS*

Penjelasan detil isi dari *problem summary* dapat dilihat pada Tabel A.1 dan Tabel A.2 Penjelasan detil dari *performance information* dapat dilihat pada Tabel A.3. Penjelasan detil dari *solution summary* dapat dilihat pada Tabel A.4 dan Tabel A.5.

**Tabel A.1 Daftar Penjelasan Isi *Problem Summary* (Bagian 1)**

| No | Nama Label | Penjelasan |
|---|---|---|
| 1. | *Objective Sense* | Tujuan obyektif (minimisasi/maksimalisasi) |
| 2. | *Objective Function* | Nama fungsi obyektif |
| 3. | *Objective Type* | Tipe obyektif (linear/nonlinear) |
| 4. | *Number of Variable* | Jumlah variabel keputusan |
| 5. | *Bounded Above* | Jumlah variable dengan batas atas saja |
| 6. | *Bounded Below* | Jumlah variable dengan batas bawah saja |
| 7. | *Bounded Below and Above* | Jumlah variable dengan batas bawah dan atas |
| 8. | *Free* | Jumlah variable tanpa memiliki batasan |
| 9. | *Fixed* | Jumlah variable dengan nilai tetap |
| 10. | *Binary* | Jumlah variable dengan tipe *binary* (0/1) |
| 11. | *Integer* | Jumlah variable dengan tipe *integer* atau bilangan bulat |
| 12. | *Number of Constraint* | Jumlah batasan |
| 13. | *Linear LE(<=)* | Jumlah batasan yang menggunakan batas bawah (<=) |
| 14. | *Linear EQ(=)* | Jumlah batasan yang tidak memakai batas / sama dengan (=) |

**Tabel A.2 Daftar Penjelasan Isi *Problem Summary* (Bagian 2)**

| No | Nama Label | Penjelasan |
|---|---|---|
| 15 | *Linear GE(>=)* | Jumlah batasan yang menggunakan batas atas (>=) |
| 16 | *Linear Range* | Jumlah batasan yang menggunakan batas atas dan batas bawah |
| 17 | *Constraint Coefficients* | Jumlah koefisien batasan |

**Tabel A.3 Daftar Penjelasan Isi *Performance Information***

| No | *Nama Label* | Penjelasan |
|---|---|---|
| 1. | *Execution Mode* | Tipe eksekusi yang digunakan |
| 2. | *Number of Threads* | Jumlah thread yang dipakai |

**Tabel A.4 Daftar Penjelasan Isi *Solution Summary* (Bagian 1)**

| No | *Nama Label* | Penjelasan |
|---|---|---|
| 1. | *Solver* | Tipe solusi (IP/MILP/NLP) |
| 2. | *Algorithm* | Algoritma yang digunakan |
| 3. | *Objective Function* | Nama fungsi obyektif |
| 4. | *Solution Status* | Status solusi (optimal/infeasible) |
| 5. | *Objective Value* | Nilai dari fungsi obyektif |
| 6. | *Relative Gap* | $\frac{\lvert \text{nilai fungsi obyektif} - best\ bound \rvert}{ie^{-10} + \lvert best\ bound \rvert}$ |
| 7. | *Absolute Gap* | $\lvert \text{nilai fungsi obyektif} - best\ bound \rvert$ |
| 8. | *Primal Infeasibillity* | Nilai maksimum pelanggaran batasan primal pada saat pencarian solusi |
| 9. | *Bound Infeasibillity* | Nilai maksimum pelanggaran pencarian solusi dari batas bawah dan batas atas |

**Tabel A.5 Daftar Penjelasan Isi *Solution Summary* (Bagian 2)**

| No | Nama Label | Penjelasan |
|---|---|---|
| 10. | *Integer Infeasibillity* | Nilai maksimum pelanggaran pencarian solusi untuk variabel bilangan bulat (*integer*) |
| 11. | *Best Bound* | Nilai terbaik dari nilai fungsi obyektif dari semua node yang diproses pada *branch and bound* pada akhir eksekusi. |
| 12. | *Nodes* | Jumlah node pada algoritma *branch and cut* |
| 13. | *Iterations* | Jumlah iterasi yang dilakukan |
| 14. | *Presolve Time* | Waktu yang digunakan untuk pre-proses. |
| 15. | *Solution Time* | Waktu yang dibutuhhkan untuk mendapatkan solusi termasuk waktu untuk pre-proses |

**a. *Screenshot* Hasil Uji 37 Pesawat (*Problem Summary, Performance Information,* dan *Solution Summary*)**

Pada bagian ini akan ditampilkan hasil screenshot hasil uji 37 pesawat untuk *problem summary, performance information,* dan *solution summary*. *Problem summary, performance information,* dan *solution summary* dapat dilihat pada Gambar A.12

**b. *Screenshot* Hasil Uji 10 Pesawat**

Pada bagian ini akan ditampilkan hasil screenshot hasil uji 10 pesawat. *Optimization result* untuk variasi kecepatan dapat dilihat pada Gambar A.1

The SAS System

The OPTMODEL Procedure

| Problem Summary | |
|---|---|
| Objective Sense | Minimization |
| Objective Function | VAC_model |
| Objective Type | Linear |
| | |
| Number of Variables | 27824 |
| Bounded Above | 0 |
| Bounded Below | 185 |
| Bounded Below and Above | 27602 |
| Free | 37 |
| Fixed | 0 |
| Binary | 27602 |
| Integer | 74 |
| | |
| Number of Constraints | 29130 |
| Linear LE (<=) | 23994 |
| Linear EQ (=) | 2435 |
| Linear GE (>=) | 2664 |
| Linear Range | 37 |
| | |
| Constraint Coefficients | 84063 |

| Performance Information | |
|---|---|
| Execution Mode | Single-Machine |
| Number of Threads | 1 |

The SAS System

The OPTMODEL Procedure

| Solution Summary | |
|---|---|
| Solver | MILP |
| Algorithm | Branch and Cut |
| Objective Function | VAC_model |
| Solution Status | Optimal within Relative Gap |
| Objective Value | 6.5562054543 |
| | |
| Relative Gap | 0.0000948696 |
| Absolute Gap | 0.0006219257 |
| Primal Infeasibility | 2.220446E-16 |
| Bound Infeasibility | 0 |
| Integer Infeasibility | 2E-6 |
| | |
| Best Bound | 6.5555835286 |
| Nodes | 420683 |
| Iterations | 2812451 |
| Presolve Time | 1.97 |
| Solution Time | 1282.98 |

**Gambar A.3 *Problem Summary*, *Performance Information*, dan *Solution Summary* Uji Coba 37 Pesawat**

| [1] | q | q_plus | q_min |
|---|---|---|---|
| 1 | 0.88152 | 0.88152 | 0.00000 |
| 2 | -1.00000 | 0.00000 | 1.00000 |
| 3 | 0.00000 | 0.00000 | 0.00000 |
| 4 | -0.89513 | 0.00000 | 0.89513 |
| 5 | 0.31355 | 0.31355 | 0.00000 |
| 6 | -0.89513 | 0.00000 | 0.89513 |
| 7 | 0.00000 | 0.00000 | 0.00000 |
| 8 | 0.31355 | 0.31355 | 0.00000 |
| 9 | -1.00000 | 0.00000 | 1.00000 |
| 10 | 0.00000 | 0.00000 | 0.00000 |

**Gambar A.4 *Optimization Result* Variabel Variasi Kecepatan Uji Coba 10 Pesawat**

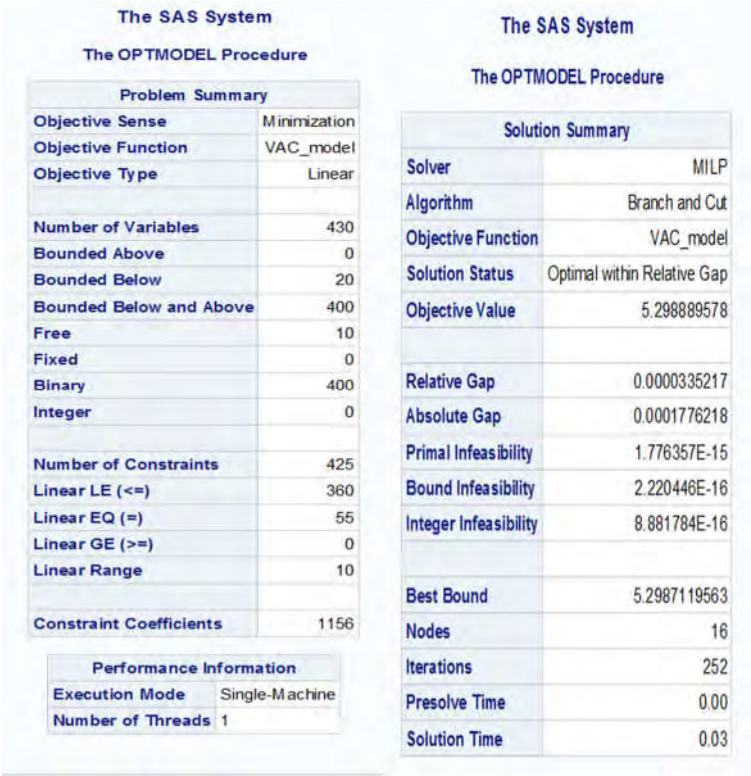
| [1] | [2] | delta_1 | delta_2 | delta_3 | delta_4 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 0 | 0 | 0 |
| 1 | 3 | 1 | 0 | 0 | 0 |
| 1 | 4 | 0 | 0 | 0 | 1 |
| 1 | 5 | 1 | 0 | 0 | 0 |
| 1 | 6 | 0 | 0 | 1 | 0 |
| 1 | 7 | 1 | 0 | 0 | 0 |
| 1 | 8 | 0 | 0 | 1 | 0 |
| 1 | 9 | 1 | 0 | 0 | 0 |
| 1 | 10 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 |
| 2 | 3 | 0 | 0 | 0 | 1 |
| 2 | 4 | 0 | 0 | 0 | 1 |
| 2 | 5 | 0 | 0 | 0 | 1 |
| 2 | 6 | 0 | 0 | 0 | 1 |
| 2 | 7 | 0 | 0 | 1 | 0 |
| 2 | 8 | 0 | 0 | 0 | 1 |
| 2 | 9 | 0 | 0 | 0 | 1 |
| 2 | 10 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 3 | 2 | 0 | 0 | 0 | 0 |
| 3 | 3 | 0 | 0 | 0 | 0 |
| 3 | 4 | 0 | 0 | 1 | 0 |
| 3 | 5 | 0 | 0 | 0 | 1 |
| 3 | 6 | 0 | 0 | 1 | 0 |
| 3 | 7 | 0 | 1 | 0 | 0 |
| 3 | 8 | 0 | 0 | 1 | 0 |
| 3 | 9 | 0 | 0 | 1 | 0 |
| 3 | 10 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 |
| 4 | 2 | 0 | 0 | 0 | 0 |
| 4 | 3 | 0 | 0 | 0 | 0 |
| 4 | 4 | 0 | 0 | 0 | 0 |
| 4 | 5 | 1 | 0 | 0 | 0 |
| 4 | 6 | 0 | 1 | 0 | 0 |
| 4 | 7 | 0 | 1 | 0 | 0 |
| 4 | 8 | 0 | 0 | 0 | 1 |
| 4 | 9 | 0 | 1 | 0 | 0 |
| 4 | 10 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 5 | 2 | 0 | 0 | 0 | 0 |
| 5 | 3 | 0 | 0 | 0 | 0 |
| 5 | 4 | 0 | 0 | 0 | 0 |
| 5 | 5 | 0 | 0 | 0 | 0 |
| 5 | 6 | 0 | 0 | 1 | 0 |
| 5 | 7 | 1 | 0 | 0 | 0 |
| 5 | 8 | 0 | 0 | 1 | 0 |
| 5 | 9 | 0 | 0 | 1 | 0 |
| 5 | 10 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 |
| 6 | 2 | 0 | 0 | 0 | 0 |
| 6 | 3 | 0 | 0 | 0 | 0 |
| 6 | 4 | 0 | 0 | 0 | 0 |
| 6 | 5 | 0 | 0 | 0 | 0 |
| 6 | 6 | 0 | 0 | 0 | 0 |
| 6 | 7 | 0 | 1 | 0 | 0 |
| 6 | 8 | 0 | 0 | 0 | 1 |
| 6 | 9 | 1 | 0 | 0 | 0 |
| 6 | 10 | 1 | 0 | 0 | 0 |

**Gambar A.5 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 1)**

| | | | | | |
|---|---|---|---|---|---|
| 7 | 1 | 0 | 0 | 0 | 0 |
| 7 | 2 | 0 | 0 | 0 | 0 |
| 7 | 3 | 0 | 0 | 0 | 0 |
| 7 | 4 | 0 | 0 | 0 | 0 |
| 7 | 5 | 0 | 0 | 0 | 0 |
| 7 | 6 | 0 | 0 | 0 | 0 |
| 7 | 7 | 0 | 0 | 0 | 0 |
| 7 | 8 | 0 | 0 | 0 | 1 |
| 7 | 9 | 0 | 0 | 0 | 1 |
| 7 | 10 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 8 | 2 | 0 | 0 | 0 | 0 |
| 8 | 3 | 0 | 0 | 0 | 0 |
| 8 | 4 | 0 | 0 | 0 | 0 |
| 8 | 5 | 0 | 0 | 0 | 0 |
| 8 | 6 | 0 | 0 | 0 | 0 |
| 8 | 7 | 0 | 0 | 0 | 0 |
| 8 | 8 | 0 | 0 | 0 | 0 |
| 8 | 9 | 0 | 1 | 0 | 0 |
| 8 | 10 | 1 | 0 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| 9 | 1 | 0 | 0 | 0 | 0 |
| 9 | 2 | 0 | 0 | 0 | 0 |
| 9 | 3 | 0 | 0 | 0 | 0 |
| 9 | 4 | 0 | 0 | 0 | 0 |
| 9 | 5 | 0 | 0 | 0 | 0 |
| 9 | 6 | 0 | 0 | 0 | 0 |
| 9 | 7 | 0 | 0 | 0 | 0 |
| 9 | 8 | 0 | 0 | 0 | 0 |
| 9 | 9 | 0 | 0 | 0 | 0 |
| 9 | 10 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 | 0 |
| 10 | 2 | 0 | 0 | 0 | 0 |
| 10 | 3 | 0 | 0 | 0 | 0 |
| 10 | 4 | 0 | 0 | 0 | 0 |
| 10 | 5 | 0 | 0 | 0 | 0 |
| 10 | 6 | 0 | 0 | 0 | 0 |
| 10 | 7 | 0 | 0 | 0 | 0 |
| 10 | 8 | 0 | 0 | 0 | 0 |
| 10 | 9 | 0 | 0 | 0 | 0 |
| 10 | 10 | 0 | 0 | 0 | 0 |

**Gambar A.6 *Optimization Result* Variabel Delta Uji Coba 10 Pesawat (Bagian 2)**



The SAS System

The OPTMODEL Procedure

| Problem Summary | |
|---|---|
| Objective Sense | Minimization |
| Objective Function | VAC_model |
| Objective Type | Linear |
| | |
| Number of Variables | 430 |
| Bounded Above | 0 |
| Bounded Below | 20 |
| Bounded Below and Above | 400 |
| Free | 10 |
| Fixed | 0 |
| Binary | 400 |
| Integer | 0 |
| | |
| Number of Constraints | 425 |
| Linear LE (<=) | 360 |
| Linear EQ (=) | 55 |
| Linear GE (>=) | 0 |
| Linear Range | 10 |
| | |
| Constraint Coefficients | 1156 |

| Performance Information | |
|---|---|
| Execution Mode | Single-Machine |
| Number of Threads | 1 |

The SAS System

The OPTMODEL Procedure

| Solution Summary | |
|---|---|
| Solver | MILP |
| Algorithm | Branch and Cut |
| Objective Function | VAC_model |
| Solution Status | Optimal within Relative Gap |
| Objective Value | 5.298889578 |
| | |
| Relative Gap | 0.0000335217 |
| Absolute Gap | 0.0001776218 |
| Primal Infeasibility | 1.776357E-15 |
| Bound Infeasibility | 2.220446E-16 |
| Integer Infeasibility | 8.881784E-16 |
| | |
| Best Bound | 5.2987119563 |
| Nodes | 16 |
| Iterations | 252 |
| Presolve Time | 0.00 |
| Solution Time | 0.03 |

**Gambar A.7 *Problem Summary*, *Performance Information*, dan *Solution Summary* Uji Coba 10 Pesawat**

*[Halaman ini sengaja dikosongkan]*

# BAB VI
# KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

## 6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap implementasi model VAC dapat diambil kesimpulan sebagai berikut:

1. Model VAC mampu menyelesaikan permasalahan penghindaran konflik karena menggunakan konstruksi geometri untuk mengatasi berbagai situasi konflik yang meliputi situasi *head to head*, *similar coordinates*, kasus anomali, *false conflict*.
2. Model VAC dapat diimplementasikan kakas bantu SAS 9.2 yang meliputi modul praproses dan modul optimasi dengan *solver Mixed Integer Linear Programming*.
3. Berdasarkan hasil uji coba, untuk uji kebenaran dengan 10 pesawat yang dilakukan oleh penulis memberikan hasil sama benarnya dengan hasil uji coba yang dilakukan oleh Alonso et al [1] dikarenakan semua konflik yang terjadi pada sebelum optimasi telah dapat dihindari. Sementara, hasil uji kinerja 37 pesawat menunjukkan nilai obyektif 6.5562 dengan jumlah *constraint* sebanyak 29130, 27602 variabel *binary*, dan 37 variabel *continue.* Dengan *cpu time* 21 menit 19,15 detik dan *real time* 21 menit 46,08 detik.

## 6.2 Saran

Beberapa saran yang hendak disampaikan terkait dengan pengerjaan tugas akhir ini adalah:

1. Diperlukan metode *pursuit situation*, yaitu jika pasangan pesawat berada dalah satu garis lurus dan salah satu pesawat memiliki batasan kecepatan yang tidak bersinggungan dengan batasan pesawat lainnya. Hal ini dapat meringkas waktu optimasi.
2. Diperlukan pengembangan model VAC terkait dengan pengembalian posisi pesawat sesuai *time flight*, sehingga pesawat dapat tiba ditujuan dengan tepat waktu.
3. Penambahan perhitungan sudut α untuk pesawat yang memiliki *safety distance* yang berbeda dengan pesawat lain.
4. Pengembangan model VAC dengan mempertimbangkan faktor luar seperti cuaca, penggunaan bahan bakar, kecepatan angin, dll.
5. Pengembangan model dengan penambahan varibel waktu dan penambahan manuver sudut arah terbang untuk mendapatkan hasil yang optimal dan media penghindaran konflik yang lengkap.
6. Penggunaan wilyah radar ATC sebagai pengganti sektor.

# DAFTAR PUSTAKA


[1] Franscisco Javier Martín-Campo, "The Collision Avoidance Problem: Methods and Algorithms," Ph.D. dissertation, Department of Statistics and Operation Research, Univ. Rey Juan Carlos, Madrid, July 2010.

[2] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 3–11, Mar. 2002.

[3] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *AIAA J. Guid., Control Dyn.*, vol. 24, no. 1, pp. 79–86, Jan./Feb. 1999.

[4] J. Krozel and M. Peters, "Strategic conflict detection and resolution for free flight," in *Proc. 36th Conf. Decision Control*, Dec. 1997, pp. 1822–1828.

[5] A. Alonso-Ayuso, L. F. Escudero, F. J. Martín-Campo, "Collision avoidance in the air traffic management: a mixed integer linear optimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 47–57, Mar. 2011.

*[Halaman ini sengaja dikosongkan]*

# BIODATA PENULIS

Izdihar Farahdina, akrab dipanggil Farah, lahir di Jakarta, pada tanggal 27 Juli 1995, merupakan anak kedua dari dua bersaudara. Penulis telah menempuh pendidikan mulai dari SD Negeri Pejompongan 05 (2001-2007), SMP Negeri 40 Jakarta (2007-2010), SMA Negeri 65 Jakarta (2010-2012), dan terakhir Teknik Informatika ITS (2012-2016). Selama kuliah, penulis pernah menjadi asisten dosen pada mata kuliah Basis Data, Pemograman Berstruktur Obyek, dan mata kuliah Kecerdasan Buatan. Dalam menyelesaikan kuliahnya, penulis mengambil bidang minat Dasar dan Terapan Komputasi (DTK) dan tertarik pada topik tugas akhir yang berhubungan dengan optimasi dan simulasi. Selain dalam bidang akademik maupun organisasi, penulis sangat suka menghabiskan waktu dengan membaca buku, jalan-jalan di alam terbuka, menonton film terutama film dengan genre *sci-fi*, mendengarkan musik, dan sebagainya. Penulis dapat dihubungi melalui e-mail: izdiharfarahdina@gmail.com.

*[Halaman ini sengaja dikosongkan]*